

DIPLOMARBEIT

Einsatz des CART-Algorithmus zur Modellierung gegnerischer Aktionen im Heads-Up Limit Texas Hold'em

von
Florian Schmidt

eingereicht am 31.07.2012 beim
Institut für Angewandte Informatik
und Formale Beschreibungsverfahren
des Karlsruher Instituts für Technologie

Referent: Prof. Dr. Hartmut Schmeck
Betreuer: Dipl.-Inform. Daniel Pathmaperuma

Heimatanschrift:
Schubertstraße 1
75365 Calw

Hiermit bestätige ich, die vorliegende Arbeit selbstständig
durchgeführt und keine anderen als die angegebenen
Literaturhilfsmittel verwendet zu haben.

Karlsruhe, 31. Juli 2012

Florian Schmidt

Ich danke

Sanaz Mostaghim

Daniel Pathmaperuma

*für Anregungen, Verbesserungsvorschläge,
stets ein offenes Ohr, moralischen Beistand
und das Verständnis für eine unkonventionelle Arbeitsweise*

meiner Mutter

für stete Unterstützung

Karlsruhe im Jahr 2012

Florian Schmidt

ZUSAMMENFASSUNG

In den vergangenen Jahrzehnten verzeichnete die Forschung große Erfolge beim Erstellen spielstarker Systeme für Spiele wie z.B. Schach, Dame oder Othello. Diese Systeme sind in der Lage, menschliche Gegner überzeugend zu schlagen. Im Gegensatz zu solchen deterministischen Spielen mit vollständiger Information, stellt das Pokerspiel die KI-Forschung vor weitaus größere und erst in Ansätzen gelöste Probleme, da es sich um ein nichtdeterministisches Spiel mit unvollständiger Information handelt, bei dem der Modellierung des Gegners eine wichtige Rolle zukommt. In dieser Arbeit wird der Einsatz des CART-Algorithmus zur Modellierung gegnerischer Aktionen in der Pokervariante Limit Texas Hold'em für zwei Spieler untersucht. Hierfür wird ein Modell zur Bereitstellung von Wahrscheinlichkeiten für die möglichen gegnerischen Aktionen entwickelt und anhand einer großen Anzahl von Spielen menschlicher Spieler analysiert. Darüber hinaus kommt das Modell in einer Miximax-Spielbaumsuche zum praktischen Einsatz und die resultierende Spielstärke dieses Gesamtsystems wird bewertet.

INHALTSVERZEICHNIS

Abbildungsverzeichnis.....	xii
Tabellenverzeichnis.....	xiii
1 Einleitung und Problemstellung.....	1
1.1 Aufgabenstellung	3
1.2 Gliederung der Arbeit.....	4
2 Grundlagen.....	5
2.1 Texas Hold'em	5
2.1.1 Eigenschaften des Pokerspiels	6
2.1.2 Spielregeln.....	7
2.1.3 Exemplarische Spielrunde	10
2.1.4 Notation des Spielverlaufs	11
2.1.5 Bestehende Ansätze	12
2.1.6 Kombination verschiedener Ansätze	21
2.2 Klassifikation.....	22
2.2.1 Entscheidungsbäume	23
2.2.2 Bewertung von Klassifikatoren.....	31
2.3 Fazit	35
3 Die Erstellung des Gegnermodells	36
3.1 Aufbau und Einsatz des Gegnermodells.....	37
3.1.1 Aufbau der Merkmalsvektoren	39
3.1.2 Weitergehende Konzepte für den Einsatz in der Praxis.....	41

3.2	Erstellung der Datenbasis.....	42
3.2.1	Struktur der Daten	44
3.3	Explorative Evaluierung der Performanz im Vergleich zu einem KNN.....	46
3.3.1	Aufbau des KNN	47
3.3.2	KKR und RMSE	47
3.3.3	Stabilität	49
3.3.4	Zeitliche Performanz	49
3.3.5	Fazit	50
3.4	Die Werkzeuge	50
3.4.1	WEKA.....	51
3.4.2	RapidMiner.....	51
3.4.3	Poker Academy Pro 2	52
3.4.4	Java/Eclipse	52
3.5	Fazit	53
4	Analyse und Bewertung des Gegnermodells	54
4.1	Maximal erreichbare Vorhersagegenauigkeit.....	54
4.1.1	KKR und RMSE nach Setzrunden	56
4.1.2	KKR und RMSE nach Spielern	56
4.1.3	Fazit	57
4.2	Anpassungsgeschwindigkeit.....	57
4.2.1	Wachsende Trainingsmenge	58
4.2.2	Konstante Größe der Trainingsmenge	61
4.2.3	Fazit	63
4.3	Klassifikationsgüte der einzelnen Aktionen	63
4.3.1	Fazit	66
4.4	Übertragbarkeit.....	66

4.4.1	Experiment Ü1: Spieler→Spieler	66
4.4.2	Experiment Ü2: Kollektiv→Spieler	69
4.4.3	Fazit	70
4.5	Zusammenfassung.....	70
5	Einsatz in der Praxis.....	71
5.1	Aufbau des Gesamtsystems: TestBot.....	71
5.2	AlwaysCall und AlwaysRaise.....	72
5.2.1	Fazit	75
5.3	Poki.....	76
5.3.1	Fazit	78
5.4	Zusammenfassung und weitere Überlegungen	78
6	Zusammenfassung und Ausblick	80
6.1	Zusammenfassung.....	80
6.2	Ausblick	82
	Literaturverzeichnis.....	84

ABBILDUNGSVERZEICHNIS

Abbildung 2.1: Informationsmenge im Texas Hold'em Spielbaum.....	15
Abbildung 2.2: Setz-Baum für die Berechnung des Erwartungswertes.....	19
Abbildung 2.3: Stark vereinfachtes Beispiel eines Klassifikationsbaums.....	28
Abbildung 3.1: Einsatz des Gegnermodells in der Spielbaumsuche.	38
Abbildung 3.2: Die Verteilung der relativen Häufigkeiten der möglichen Aktionen	46
Abbildung 3.3: Vergleich der Klassifikationsleistung von CART zu der eines KNN	48
Abbildung 4.1: Lernkurve (KKR) bei wachsender Größe der Trainingsmenge für Spieler8	59
Abbildung 4.2: Lernkurve (RMSE) bei wachsender Größe der Trainingsmenge für Spieler8 ..	60
Abbildung 4.3: Lernkurve (KKR) bei wachsender Größe der Trainingsmenge für Spieler7	61
Abbildung 4.4: RMSE100-Linien für verschiedene Größen der Trainingsmengen	62
Abbildung 5.1: Eine Serie von 8317 Spielen gegen die Benchmark-Strategie AlwaysCall.....	74
Abbildung 5.2: Eine Serie von 3524 Spielen gegen die Benchmark-Strategie AlwaysRaise. ...	75
Abbildung 5.3: Eine Serie von 18.826 Spielen gegen Poki	76

TABELLENVERZEICHNIS

Tabelle 1: Wertigkeit und Wahrscheinlichkeit der verschiedenen Hände.	9
Tabelle 2: Übersicht und Erklärung der verwendeten Merkmale	39
Tabelle 3: Anzahl der Spiele und Entscheidungspunkte für die hier betrachteten Spieler.	44
Tabelle 4: KKR und RMSE für alle Spieler und Setzrunden	55
Tabelle 5: Übersicht über die Sensitivität und den PPV für die drei Zielklassen	64
Tabelle 6: KKR und RMSE einer abgewandelten Kreuzvalidierung (Spieler→Spieler)	67
Tabelle 7: KKR und RMSE für eine 10-fache Kreuzvalidierung (Kollektiv→Spieler)	69

1 EINLEITUNG UND PROBLEMSTELLUNG

„Probably a truly intelligent machine will carry out activities which may best be described as self-improvement. Some schemes for doing this have been proposed and are worth further study. It seems likely that this question can be studied abstractly as well.“

A Proposal for the Dartmouth Summer Research Project on Artificial Intelligence, 31.August 1955 [MMRS06]

Spiele stellen seit Urzeiten eine Herausforderung dar und der Umgang bzw. die Analyse von Spielen war schon immer eine Quelle neuer Erkenntnisse. Prinzipien, die in einem Spiel erkannt wurden, konnten sich oftmals leicht auf das „wirkliche Leben“ bzw. Realweltprobleme übertragen lassen. Seit dem Aufkommen von Computern gibt es eine ganz neue Art von Herausforderung. Diese besteht darin, Verfahren und Algorithmen zu entwickeln, die Computer in die Lage versetzen sollen, alle Arten von Spielen zu meistern, d.h. besser zu spielen als die besten menschlichen Spieler. Grundlegende Arbeiten hierzu sind schon fast so alt, wie der Computer selbst. Beispielsweise beschäftigte sich Norbert Wiener, der als Begründer der Kybernetik gilt, bereits 1948 damit, wie man mittels einer Minimax-Baumsuche und geeigneter Heuristiken einen Computer in die Lage versetzen könnte, auf einem annehmbaren Niveau Schach zu spielen [Wie61]. Das erste Programm, das offiziell einen Weltmeistertitel gegen menschliche Gegner errang, war CHINOOK, das 1994 den amtierenden Weltmeister im Dame-Spiel, Marion Tinsley schlug [Sch97]. Seitdem wurden die Verfahren zur Erstellung von spielstarken Computer-Programmen stetig weiterentwickelt und verfeinert, sodass sich die besten menschlichen Spieler in immer mehr Spielen dem Computer geschlagen geben mussten. Zu erwähnen ist hier das Programm Logistello, das 1997 den amtierenden Othello-Weltmeister mit 6:0 besiegte [Bur97]. Im selben Jahr unterlag Schachweltmeister Garry Kasparov in einem kurzen Schaukampf dem äußerst spielstarken Schachprogramm Deep Blue knapp mit 3,5 zu 2,5 [CHH02].

2005 gewann dann das Programm HYDRA gegen Michael Adams - einen der stärksten Schachspieler der Welt - mit fünf Siegen, einem Unentschieden und keiner Niederlage. Dies verdeutlicht die schnell verlaufende Entwicklung der Verfahren und Algorithmen aus den Bereichen der Spieltheorie auf der einen Seite und der künstlichen Intelligenz auf der anderen. Alle bisher genannten Spiele haben eine Gemeinsamkeit: Es gibt für die Spieler keine unbekannten Informationen, wie etwa verdeckte gegnerische Karten, und insbesondere keine Zufallselemente, wie das Werfen eines oder mehrerer Würfel. Für diese Spiele ist in der Regel eine der verschiedenen Minimax-Varianten, gepaart mit guten Heuristiken zur Bewertung von Spielpositionen, ausreichend, um Programme zu erstellen, die auch die besten menschlichen Spieler übertreffen. Im Schachspiel verfügen diese Programme zusätzlich noch über Eröffnungs- und Endspiel-Datenbanken. Das Problem dieser Art von Spielen ist prinzipiell nur der hohe Verzweigungsfaktor der zugehörigen Spielbäume. Im Falle des Go-Spiels ist der Verzweigungsfaktor so groß, dass traditionelle Minimax-Ansätze bisher noch keine Programme hervorgebracht haben, die es mit menschlichen Großmeistern aufnehmen können.

Eine weitere Klasse von Spielen verfügt darüber hinaus über ein Zufallselement, beispielsweise den Wurf eines oder mehrerer Würfel oder das Austeilen einer oder mehrerer Karten. Für diese Klasse von Spielen vollständiger Information existieren Abwandlungen der Minimax-Suche, beispielsweise der Expectimax-Algorithmus. Auch Ansätze, die auf Monte-Carlo-Simulationen beruhen, führten zu äußerst spielstarken Programmen.

Die größte Herausforderung stellen allerdings Spiele dar, die neben einem Zufallselement auch noch die Eigenschaft der unvollständigen Information besitzen, d.h. neben Zufallselementen gibt es Informationen, die nicht allen Spielern zugänglich sind. Ein bekannter Vertreter dieser Klasse von Spielen ist das Pokerspiel. Seit etwa 15 Jahren erfährt es eine immer größer werdende Aufmerksamkeit seitens der Forschung. Die Pokervariante, der dabei die größte Aufmerksamkeit zuteilwird, ist Texas Hold'em, das gemeinhin als die einfachste, aber auch die strategisch komplexeste bekannt ist. Um erst einmal grundlegende Techniken zu entwickeln, wird dabei häufig – und insbesondere in dieser Arbeit – die Zwei-Spieler-Variante betrachtet.

Die Entwicklung der Pokersysteme¹ begann bei einfachen regel- und formelbasierten Expertensystemen, führte über Simulationstechniken und spieltheoretische Lösungen bis hin zu einer Erweiterung der klassischen Spielbaumsuche für Spiele mit unvollständiger Information und Zufallselementen. Das Herzstück dieser erweiterten Spielbaumsuche ist das Gegnermodell. Es liefert an den gegnerischen Entscheidungsknoten und an den Spielendknoten Wahrscheinlichkeiten für die möglichen gegnerischen Aktionen bzw. Gewinnwahrscheinlichkeiten, aus denen sich dann der Erwartungswert für diese Knoten errechnen lässt. Je genauer ein solches Gegnermodell ist, desto genauer können auch die Erwartungswerte für die eigenen Aktionen abgeschätzt werden.

Aus dieser Situation heraus leitet sich die Aufgabenstellung ab, die dieser Arbeit zugrunde liegt.

1.1 AUFGABENSTELLUNG

Die Aufgabenstellung dieser Arbeit ist die Erstellung eines Modells zur Vorhersage gegnerischer *Post-Flop* Aktionen² unter Einsatz des CART-Algorithmus³ für die Pokervariante *Heads-Up Limit Texas Hold'em*⁴ und dessen Evaluierung.

Die Evaluierung umfasst dabei zwei Unteraufgaben. Zum Einen die Analyse der Klassifikationsleistung des auf das Spezialgebiet Poker angewendeten CART-Algorithmus. Insbesondere soll hier die Frage beantwortet werden, ob sich Aussagen darüber treffen lassen, wie schnell CART „hinreichend gute“ Ergebnisse liefern kann und welche Performanz sich im optimalen, d.h. im „ausgelernten“, Fall erreichen lässt.

Zum Anderen soll das hier entwickelte Gegnermodell in der Miximax-Baumsuche zum praktischen Einsatz gebracht werden und gegen verschiedene Computer-Pokersysteme antreten.

1 Mit dem Begriff Pokersystem wird in dieser Arbeit allgemein ein Programm bzw. ein Agent bezeichnet, das/der in der Lage ist, den Regeln entsprechend Poker zu spielen.

2 Dies sind alle Aktionen, die nach dem Austeilen der ersten drei Gemeinschaftskarten erfolgen.

3 CART ist ein Klassifikationsalgorithmus, der zu den Entscheidungsbäumen zählt. Der Name CART leitet sich aus dem Titel des Buches „Classification And Regression Trees“ von Breiman et al. ab [BFOS84].

4 Heads-Up bedeutet, dass das Spiel von ausschließlich zwei Spielern gespielt wird.

1.2 GLIEDERUNG DER ARBEIT

In Kapitel 2 erfolgt zunächst eine Einführung in die Grundlagen des Pokerspiels und der Klassifikation, die zum Verständnis der weiteren Kapitel Voraussetzung sind. Insbesondere wird hier auch auf bestehende Ansätze zur Erstellung von Pokersystemen eingegangen, da dies für eine Einordnung der vorliegenden Arbeit unabdingbar ist.

Kapitel 3 stellt das hier entwickelte Gegnermodell⁵ und die verwendeten Werkzeuge vor, bevor dieses in Kapitel 4 anhand realer Daten evaluiert wird.

Im 5. Kapitel kommt das Modell in einer Miximax-Baumsuche zum Einsatz, und es wird ein Blick auf die resultierende Spielstärke geworfen. Kapitel 6 liefert eine Zusammenfassung der vorliegenden Arbeit und geht auf die Möglichkeiten zukünftiger Entwicklungen ein.

Dieser Arbeit liegt eine CD-ROM bei, die alle hier verwendeten Daten und Werkzeuge, inklusive der hier dargestellten Experimente, enthält. Außerdem findet sich auf ihr, bis auf wenige Ausnahmen, die referenzierte Literatur.

⁵ Prinzipiell stellt das hier entwickelte Modell eine Teilkomponente eines Gegnermodells dar, da hier die Vorhersage der gegnerischen Aktionen im Mittelpunkt steht. Ein komplettes Gegnermodell umfasst zusätzlich die Ermittlung der Erwartungswerte an den Blattknoten des Spielbaumes. Im Verlauf dieser Arbeit wird diese Teilkomponente trotzdem gemeinhin als Gegnermodell bezeichnet.

2 GRUNDLAGEN

„Artificial intelligence is knowing what to do when we don't know what to do.“

Peter Norvig

In Bezug auf das Pokerspiel liegt der Schwerpunkt der Forschung auf der Variante des Texas Hold'em. Die Regeln sind im Vergleich zu anderen Pokervarianten sehr einfach, jedoch gilt Texas Hold'em nach Billings [Bil06] als die Pokervariante, bei der es am meisten auf die Fähigkeiten der einzelnen Spieler ankommt. In diesem Kapitel werden zunächst die Regeln und Eigenschaften des Limit Hold'em Spiels erläutert, bevor ein Blick auf bestehende Ansätze zur Erstellung von Computer-Programmen für diesen Bereich geworfen wird. Im Anschluss daran werden einige Grundbegriffe aus dem Bereich der Klassifikation eingeführt, soweit sie für das weitere Verständnis der Arbeit von Interesse sind.

2.1 TEXAS HOLD'EM

Texas Hold'em wird in mehreren Varianten gespielt. Die populärsten beiden sind zum Einen *No Limit Texas Hold'em*, bei dem die Einsätze der einzelnen Spieler ausschließlich durch die Anzahl der eigenen Chips begrenzt sind, und zum Anderen *Limit Texas Hold'em*⁶, bei dem die Einsätze und Erhöhungen immer nur fixe Beträge zulassen. Bisher war meist nur die Limit-Variante Gegenstand der Forschung, da es zunächst darum geht, grundlegende Verfahren und Algorithmen zu entwickeln, einzusetzen und zu erproben. Das No Limit-Spiel ist im Vergleich zum Limit Texas Hold'em weitaus komplexer. Dies ist der Tatsache geschuldet, dass die Höhe der Einsätze bzw. der Erhöhungen nicht festgesetzt ist⁷.

6 Limit Texas Hold'em wird oft auch als Fixed Limit Hold'em bezeichnet.

7 Die Höhe eines Einsatzes muss mindestens der Höhe des Big Blind entsprechen. Eine Erhöhung muss mindestens das Doppelte des zuvor getätigten Einsatzes (bzw. der zuvor getätigten Erhöhung) betragen.

2.1.1 Eigenschaften des Pokerspiels

Verschiedene Varianten des Pokerspiels wurden bereits seit den zwanziger Jahren des vergangenen Jahrhunderts theoretisch durchleuchtet und als Beispiele für spieltheoretische Prinzipien und Anwendungen herangezogen. Alan Turing, John von Neumann und John Forbes Nash Jr. sind nur die bekanntesten Namen, die in diesem Zusammenhang genannt werden müssen. Wie die meisten anderen Spiele auch, besitzt das Pokerspiel einige Eigenschaften, die es zu einem idealen Forschungsobjekt machen (siehe [Bil06]):

- Einfache, klar definierte Regeln. Im Falle des 2-Spieler Limit Hold'em hat ein Spieler je nach Situation nur zwei oder drei verschiedene Handlungsalternativen.
- Das Ziel des Spiels ist klar definiert.
- Der Ausgang eines Spiels ist beobachtbar. Am Ende eines Spiels gibt es einen Gewinner und einen Verlierer. Eine Serie von Spielen erlaubt eine Aussage über den Erfolg bzw. die Spielstärke verschiedener Strategien.

Im Gegensatz zu Spielen mit vollständiger Information wie Schach, Dame oder Backgammon, verfügt das 2-Spieler Texas Hold'em Spiel darüber hinaus über Eigenschaften, die für die Forschung zusätzliche Herausforderungen darstellen:

- **Unvollständige Information**, d.h. unbekannte gegnerische Karten, erfordert eine Modellierung des Gegners. Das Täuschen des Gegenspielers hinsichtlich der eigenen Handstärke und das Erkennen gegnerischer Täuschungsversuche sind wesentlicher Bestandteil des Spiels.
- **Zufallseignisse**, d.h. das Austeilen zufälliger Karten, induzieren Unsicherheit hinsichtlich der zukünftigen Entwicklung der Stärke des eigenen Blattes. Ein hohes Maß an Varianz macht die Analyse der Spielstärke verschiedener Strategien äußerst schwierig und erfordert die Durchführung einer großen Anzahl an Spielen.

Abgesehen davon, kann für ein Setzen oder eine Erhöhung jeder beliebige Betrag, bis zur Größe des eigenen Chip-Stapels, gewählt werden.

- Die Spielzustände sind nur **teilweise beobachtbar**. Informationen über die gegnerischen Karten lassen sich nur am Ende eines Spiels erlangen, wenn die Karten aufgedeckt werden und der Gewinner ermittelt wird. Entschließt sich der Gegner im Laufe eines Spiels dazu, seine Karten wegzuerwerfen, so ist das Spiel beendet und ein Informationsgewinn findet nicht statt.

2.1.2 Spielregeln

Texas Hold'em ist ein Spiel, das mit einem handelsüblichen Kartenspiel von 52 Karten gespielt wird. Üblicherweise nehmen 2 bis 11 Spieler an einem Spiel teil. Zu Beginn jedes Spiels, auch *Hand*⁸ genannt, werden jedem Spieler zwei Karten, die sogenannten *Hole Cards*, verdeckt ausgeteilt, die im weiteren Verlauf nur ihm bekannt sind. Das Ziel des Spiels besteht darin, mit den Hole Cards und den fünf Karten, die nach und nach auf dem Tisch ausgeteilt werden, die stärkste 5-Karten-Kombination zu erzielen und den sogenannten *Pot*, das heißt die von allen Spielern während des Spiels getätigten Einsätze, zu gewinnen. Interessant hierbei ist, dass ein Spieler durch geschicktes Setzen den Pot auch mit einer Hand gewinnen kann, die nicht die stärkste Hand aller Mitspieler ist, indem er *blufft*, d.h. die anderen Spieler über die Stärke seiner tatsächlichen Hand täuscht.

Vor Beginn des ersten Spiels werden alle 52 Karten gemischt, und jedem Spieler wird eine Karte ausgeteilt. Dem Spieler mit der höchsten⁹ Karte wird der *Dealer Button* zugeteilt, der anzeigt, wer die Karten gibt. In einem Casino übernimmt ein Croupier das Austeilen der Karten, der Dealer-Button wird jedoch trotzdem für die sogenannten *Blinds*¹⁰ benötigt. Im Folgenden wird nur das *Heads-Up* Spiel betrachtet, d.h. das Spiel mit zwei Spielern. Zu Beginn eines Spiels setzen beide Spieler die Blinds. Der Spieler, der den Dealer Button besitzt

8 Der Begriff Hand wird oft synonym für ein Spiel gebraucht. Darüber hinaus wird auch die beste Kartenkombination, die ein Spieler bilden kann als Hand bezeichnet. Insbesondere vor dem Flop werden die beiden Hole Cards als Starthand bzw. „starting hand“ bezeichnet.

9 Die Wertigkeiten der Karten in aufsteigender Reihenfolge: 2, 3, 4, 5, 6, 7, 8, 9, T (Zehn), J (Bube), Q (Dame), K (König), A (Ass). Die Farben (Pik, Kreuz, Karo, Herz) haben keinen Einfluss auf die Wertigkeit einer Karte.

10 *Blinds* sind am Tisch zuvor festgelegte Pflichteinsätze, die vor Beginn des Spiels zu entrichten sind.

- im Folgenden SP1 genannt - setzt einen halben *Small Bet*¹¹, der Spieler ohne Dealer-Button
- SP2- einen vollen Small Bet.

Nun folgt das Austeilen der Karten. Jeder Spieler erhält vom Geber zwei Karten, die nur ihm bekannt sind und legt diese verdeckt vor sich auf den Tisch. Daraufhin folgt die erste Setzrunde, die SP1 beginnt. Da dieser bisher nur einen halben Small Bet gesetzt hat, muss er nun entscheiden, ob seine beiden Hole Cards gut genug sind, um entweder seinen bereits geleisteten halben Small Bet auf einen vollen Small Bet aufzustocken (*Call*) oder aber sogar eine Erhöhung zu tätigen (*Raise*). Sollten die ausgeteilten Karten keinen Call oder Raise rechtfertigen, so kann er seine Karten auch wegwerfen (*Fold*) und so dieses Spiel beenden. Der Dealer Button wechselt dann zu SP2, der nun seinerseits nur einen halben Small Bet entrichten muss, und nach dem Mischen der Karten beginnt ein neues Spiel. Prinzipiell stehen einem Spieler, der am Zug ist, immer genau drei Möglichkeiten zur Verfügung: *schieben/mitgehen*, *setzen/erhöhen* oder *aussteigen*. Es kann immer nur dann geschoben werden, wenn in der aktuellen Setzrunde noch kein Spieler erhöht hat. Schieben ist das Weitergeben an den nächsten Spieler ohne Einsatz. Entscheidet sich der Spieler zum Setzen, so kann er beim Limit Texas Hold'em immer nur einen fixen Betrag setzen, und zwar den Small Bet in den ersten beiden Setzrunden, und den doppelten Small Bet, auch Big Bet genannt, in den letzten beiden Setzrunden. Erhöhungen sind nur möglich, wenn in der aktuellen Setzrunde bereits gesetzt wurde. Eine Erhöhung kann auch immer nur in Höhe des Small Bets, respektive des Big Bets erfolgen, allerdings ist die Anzahl der Erhöhungen für alle Spieler pro Setzrunde auf drei beschränkt (bei vorherigem Setzen eines Spielers). Der Pot kann folglich in jeder Setzrunde höchstens um folgenden Betrag wachsen:

$$(\text{Anzahl der aktiven Spieler}) \times 4 \times \text{Einsatz} \quad (2.1)$$

Einsatz bedeutet hier wieder entweder der Small Bet oder der Big Bet. Das Mitgehen eines Spielers setzt voraus, dass er den von seinem Gegner getätigten Einsatz bzw. dessen

¹¹ Die Größe des *Small Bet* variiert üblicherweise zwischen 2 Cent und mehreren hundert Dollar. Es ist anzumerken, dass der Small Blind einem halben Small Bet entspricht. Der Big Blind entspricht einem vollen Small Bet.

Erhöhung bezahlt. Erscheint einem Spieler seine Hand nicht hinreichend stark, als dass sie ein Mitgehen, Setzen oder Erhöhen rechtfertigt, so kann er aus dem Spiel aussteigen. Er verliert damit das Spiel und hat keine Chance mehr, den Pot zu gewinnen.

Auf die erste Setzrunde, die auch *Preflop* genannt wird, folgt das Austeilen dreier Karten, des sogenannten *Flop*. Diese drei Karten liegen in der Mitte des Spieltisches und sind für alle Spieler zu sehen. Nun folgt die zweite Setzrunde. Abgesehen von der ersten Setzrunde werden alle weiteren Runden von dem Spieler begonnen, der auch initial den vollen Small Bet entrichtet hat. Beachtenswert ist, dass jetzt erstmals 5-Karten-Kombinationen der eigenen Hand mit den öffentlichen Karten, die auch *Board Cards* oder *Community Cards* genannt werden, gebildet werden können. Das bedeutet, dass die Hand eine neue Wertigkeit in Bezug auf die Chancen bekommt, das Spiel am Ende zu gewinnen. Nach dem Ende der zweiten Setzrunde wird eine weitere Karte, die sogenannte *Turn-Karte* aufgedeckt und eine erneute Setzrunde folgt. Ist diese beendet, wird die letzte Karte, die *River-Karte*, aufgedeckt, und eine finale Setzrunde folgt, an die sich dann der sogenannte *Showdown*, das Aufdecken der bis dahin verdeckten Hände, anschließt, und der Pot dem Spieler mit der stärksten Hand zugeteilt wird. Beachtenswert ist jedoch, dass eine Spielrunde auch ohne Showdown, d.h. dem Vergleichen der Hole Cards, beendet werden kann¹².

Die Wertigkeiten aller möglichen 5-Karten-Kombinationen und deren Wahrscheinlichkeiten liefert Tabelle 1. Die Wahrscheinlichkeiten beziehen sich darauf, eine bestimmte Hand am Ende des Spiels zu erzielen, noch bevor die Hole Cards ausgeteilt wurden. Die Wertigkeit der Hände, die in der Tabelle aufgeführt sind, nimmt von oben nach unten zu, während die Wahrscheinlichkeiten, eine bestimmte Hand zu erzielen, abnehmen.

Tabelle 1: Wertigkeit und Wahrscheinlichkeit der verschiedenen Hände.

Hand	Beispiel	Wahrscheinlichkeit
High Card	A♥ T♠ 2♦ 6♣ 7♠ J♦ 8♦	0,1741192
Paar	A♥ A♠ 2♦ 6♣ 7♠ J♦ 8♦	0,438322546
Zwei Paare	A♥ A♠ 2♦ 2♣ 7♠ J♦ 8♦	0,23495536

¹² Insbesondere in der 2-Spieler Variante ist ein Showdown eher die Ausnahme. Dies hat negative Auswirkungen auf die Geschwindigkeit (d.h. die benötigte Anzahl an beobachteten Spielen), mit der ein Gegnermodell akzeptable Wahrscheinlichkeiten zur Approximation der Erwartungswerte an den Spielendknoten liefern kann.

Hand	Beispiel	Wahrscheinlichkeit
Drilling	A♥ A♠ A♦ 6♣ 7♠ J♦ 8♦	0,0482987
Straße	J♥ T♠ 2♦ 3♣ 4♠ 5♦ 6♦	0,0461938
Flush	A♥ T♥ 2♥ 6♥ 7♥ J♣ 8♣	0,0303255
Full House	A♥ A♠ 2♣ 2♥ 2♠ J♣ 8♣	0,02596102
Vierling	A♥ T♥ 2♣ 2♥ 2♠ 2♦ 8♣	0,00168067
Straight Flush	5♥ 6♥ 7♥ 8♥ 9♥ 2♦ 8♣	0,00027851
Royal Flush	T♥ J♥ Q♥ K♥ A♥ 2♦ 8♣	0,00003232

2.1.3 Exemplarische Spielrunde

Stellen wir uns ein Texas Hold'em Spiel mit 2 Spielern vor, SP1 und SP2. SP1 hält den Dealer Button, SP1 muss also folglich einen halben Small Bet, SP2 einen vollen Small Bet entrichten. Daraufhin werden die Karten ausgeteilt. SP1 erhält K♣7♠, SP2 8♥9♥. SP1 stockt seinen halben Small Bet auf den vollen Small Bet auf und SP2 schiebt. Die erste Setzrunde ist zu Ende und der Flop wird ausgeteilt: K♠ 7♥ 10♥.

SP1 hält nun mit zwei Paaren eine starke Hand, die momentan auch noch die stärkste am Tisch ist. Die Hand von SP2 hat mit der höchsten Karte König eine sehr geringe Wertigkeit, jedoch ein sehr hohes Potenzial sich zu verbessern, da jedes weitere Herz oder aber eine 6 oder ein Bube ihm einen Flush oder eine Straße und somit sehr wahrscheinlichen den Sieg einbringen werden. SP2 rechnet sich aus, dass er 47 Karten nicht kennt, davon neun Herz, vier Buben und vier Sechser. Da nach dem Flop noch 2 weitere Karten aufgedeckt werden, entscheidet SP2, dass seine Chancen auf den Sieg hinreichend hoch sind und setzt einen Small Bet. SP1, sehr zufrieden mit seinen 2 Paaren, welche durch jeden weiteren König oder jede weitere Sieben zum Full House werden, entschließt sich zu einer Erhöhung um einen Small Bet. SP2 geht mit und die Turn-Karte wird aufgedeckt: A♥. SP2 hat nun einen Flush und entschließt sich, die Stärke seiner Hand zu verschleiern und schiebt. SP1, dem die Turn-Karte nicht geholfen hat, ist sich noch immer sicher, dass seine Hand trotz dreier Herzen auf dem Tisch die stärkste ist, und setzt, wie für die letzten beiden Setzrunden erforderlich, einen Big Bet, den SP2, um den Eindruck einer schwächeren Hand zu erwecken, nach längerem Überlegen mitgeht. Nun wird die finale River-Karte aufgedeckt: 4♣. SP2 setzt daraufhin einen Big Bet, was SP1 nach dem Schieben und zögerlichen Mitgehen für einen Bluff hält, und erhöht. SP2 erhöht nun seinerseits, woraufhin SP1 wiederum erhöht. SP2 geht diese

Erhöhung mit, könnte auch selbst gar nicht mehr erhöhen, da die Anzahl der Erhöhungen auf drei pro Setzrunde begrenzt ist. Nun kommt es zum Showdown: Beide Spieler decken ihre Karten auf und den beiden Paaren von SP1 steht ein Flush von SP2 gegenüber, der damit 22 Small Bets gewinnt.

Diese exemplarische Spielrunde verdeutlicht verschiedene Aspekte des Pokerns, die auch für die Konstruktion eines spielstarken Pokersystems von Bedeutung sind. Zum Einen muss solch ein System die Stärke der eigenen Hand aber auch deren Potenzial in Bezug auf weitere Gemeinschaftskarten kennen. Zum Anderen sind Aspekte wie das Verschleiern der eigenen Handstärke und Bluffen wichtige Verhaltensweisen, ohne die ein Pokersystem nicht die Spielstärke von Weltklasse-Spielern erreichen kann.

2.1.4 Notation des Spielverlaufs

An einigen Stellen in dieser Arbeit wird eine Notation verwendet, um den Verlauf eines Spieles hinsichtlich des Setzverhaltens zu beschreiben. Dabei werden die einzelnen Aktionen der beiden Spieler folgendermaßen abgekürzt:

- f: fold (aussteigen)
- c: call (mitgehen)
- k: check (schieben)
- b: bet (setzen)
- r: raise (erhöhen)

Die Aktionen des zu modellierenden Spielers werden in Kleinbuchstaben dargestellt, wohingegen die Aktionen seines Gegners in Großbuchstaben angegeben werden. Die einzelnen Setzrunden werden durch einen senkrechten Strich voneinander getrennt. Die Blinds werden in dieser Notation nicht aufgeführt, da sie aus der ersten Aktion abgeleitet werden können¹³. Gemäß diesen Konventionen können die Aktionen beider Spieler vollständig beschrieben werden. Ein kurzes Beispiel verdeutlicht die Notation: *cK/BrC/Bf*.

¹³ Der Spieler mit dem Dealer Button sitzt im Small Blind, d.h. er entrichtet einen halben Small Bet und beginnt die erste Spiel- bzw. Setzrunde.

2.1.5 Bestehende Ansätze

Es ist sicherlich schwierig den Beginn der Forschung bezüglich der Entwicklung von Pokerprogrammen genau zu datieren. Dennoch soll hier auf eine kurze Darstellung einer Entwicklung der verschiedenen Ansätze nicht verzichtet werden, da sie für die Einordnung des in dieser Arbeit vorgestellten Modells hilfreich ist.

Bereits in den 60er-Jahren des vergangenen Jahrhunderts widmete sich Nicholas Findler der Erstellung eines Computerprogramms für das Pokerspiel [Fin61]. Der Schwerpunkt seiner Arbeiten lag dabei auf der Modellierung kognitiver Prozesse und das resultierende Programm war nicht sehr spielstark. Wirklich spielstarke Programme wurden erstmals durch die 1997 gegründete *Computer Poker Research Group* (CPRG) an der Universität von Alberta, Kanada, entwickelt. Im Folgenden werden die verschiedenen Ansätze der CPRG kurz, chronologisch geordnet, beschrieben. Im Folgenden werden die wesentlichen Merkmale stark vereinfacht dargestellt. Einen umfassenden Überblick über die erläuterten Ansätze liefert Billings in [Bil06].

2.1.5.1 WISSENSBASIERTE METHODEN UND SIMULATION

Die frühen Arbeiten der CPRG beschäftigten sich mit der Konstruktion eines Programms für das Texas Hold'em Spiel, das unter dem Namen LOKI [Pap98] und später Poki [Dav02] sehr bekannt wurde. Im Gegensatz zu späteren Arbeiten wurde dieses System nicht für die Heads-Up Variante entwickelt, sondern für das Spielen gegen mehrere Gegner.

Zu Beginn wurde zunächst ein formelbasiertes System konstruiert, das aus Faktoren wie z.B. der Stärke und den Entwicklungsmöglichkeiten der eigenen Hand und der durch ein Gegnermodell ermittelten wahrscheinlichen gegnerischen Karten die nächste eigene Aktion berechnete. Später wurde der formelbasierte Ansatz aufgrund der Beschränktheit und der mangelnden Spielstärke zugunsten eines simulationsbasierten Ansatzes aufgegeben, d.h. es wurden, ausgehend vom augenblicklichen Spielzustand, viele Spiele für jede eigene mögliche Aktion simuliert. Ein Gegnermodell war dafür zuständig, Wahrscheinlichkeiten für die zu simulierenden gegnerischen Aktionen und mögliche gegnerische Karten zu liefern.

2.1.5.2 SPIELTHEORETISCHE METHODEN

Nachdem John von Neumann und John Nash¹⁴ das Pokerspiel bzw. vereinfachte Pokervarianten zur Illustration ihrer Theorien benutzt hatten, war es dennoch lange Zeit unmöglich gewesen, spieltheoretische Verfahren anzuwenden um Gleichgewichtsstrategien zu berechnen. Dies lag vor allem daran, dass der zum Texas Hold'em für zwei Spieler gehörige Spielbaum eine Größe von ungefähr 10^{18} besitzt¹⁵. In [BBDS03] wird gezeigt, wie sich das Problem durch geschickte Abstraktionen und Zerlegung des Spiels in zwei kleinere Probleme der Größe von jeweils 10^7 zerlegen lässt, für die dann Nash-Gleichgewichte in gemischten Strategien berechnet werden konnten. Eine anschließende Rückübertragung dieser Strategien auf das richtige Spiel hatte eine ganze Familie von äußerst spielstarken Programmen zur Folge. Problematisch erwies sich hier, dass das Zusammenfügen der beiden abstrakten Lösungen nicht notwendigerweise selbst eine Gleichgewichtsstrategie darstellt.

Ein Vertreter dieser Familie, *PSOPT11*, unterlag Gautam Rao, einem Weltklasse-Spieler, im Verlauf einer Serie von 7030 Spielen nur knapp.

Prinzipiell stellt eine spieltheoretische Lösung, insbesondere im Bereich des Pokerspiels, eine defensive Strategie dar. Dem Gegner wird im Grunde perfektes Spielen unterstellt und der maximale eigene Verlust wird minimiert. In Bezug auf das Pokerspiel ist diese Eigenschaft zugleich als Stärke aber auch als Schwäche zu bewerten. Die Stärke ist, dass eine Gleichgewichtsstrategie auf lange Sicht nicht geschlagen werden kann, sie garantiert den

14 Nachdem John Nash in seiner 27-seitigen Dissertation von 1950 seine Theorie der *Gleichgewichtspunkte (equilibrium points)* darlegt und anschließend an einem stark vereinfachten Pokerspiel für 3 Personen verdeutlicht, beschließt er seine Arbeit mit einem Abschnitt über die Anwendungsmöglichkeiten seiner Theorie:

„The study of n-person games for which the accepted ethics of fair play imply non-cooperative playing is, of course, an obvious direction in which to apply this theory. And poker is the most obvious target. The analysis of a more realistic poker game than our very simple model should be quite an interesting affair. The complexity of the mathematical work needed for a complete investigation increases rather rapidly, however, with increasing complexity of the game; so that analysis of a game much more complex than the example given here might only be feasible using approximate computational methods [...]”.

Im Grunde stellt der spieltheoretische Ansatz der CPRG eine Umsetzung der von Nash erwähnten „*approximate computational methods*“ dar.

¹⁵ Die Anzahl aller möglichen Spiele beträgt exakt 1.179.000.604.565.715.751

spieltheoretischen Wert des Spiels. Auf der anderen Seite ist es möglich, dass äußerst schlechte Spieler nur ganz knapp geschlagen werden. Offensichtliche Schwächen im Spiel des Gegners werden ignoriert. Explizit anzumerken ist, dass eine spieltheoretische Lösung kein Gegnermodell beinhaltet. Erwähnenswert ist weiterhin die Tatsache, dass spieltheoretische Lösungen, die das richtige Spiel besser approximieren, im Normalfall stärker spielen als Lösungen, die stärker abstrahieren [WBB09].

2.1.5.3 ADAPTIVE SPIELBAUMSUCHE FÜR SPIELE IMPERFEKTER INFORMATION

Da in dieser Arbeit ein Gegnermodell entwickelt werden soll, das in verschiedenen Varianten der Spielbaumsuche eingesetzt werden kann, wird die von der CPRG entwickelte und in [BDSB04] vorgestellte Variante der adaptiven Spielbaumsuche in den Ausprägungen des Miximax- bzw. Miximax-Algorithmus hier ausführlich dargestellt.

Die Ergebnisse hinsichtlich des spieltheoretischen Ansatzes der CPRG waren zwar sehr ermutigend, doch erstens waren die resultierenden Strategien nur grobe Annäherungen an eine wirkliche Nash-Gleichgewichtslösung, und zweitens sind solche Strategien problematisch, da sie zwar, sofern es wirkliche Gleichgewichtslösungen sind und nicht nur Annäherungen, auf lange Sicht nicht zu schlagen sind, jedoch dem Gegner perfektes Spiel unterstellen und nicht nach Schwächen in der gegnerischen Strategie suchen und diese ausnutzen. Doch gerade das Einstellen auf die Strategie des Gegners und das Ausnutzen sub-optimalen Spielens entscheidet beim Texas Hold'em oft über Sieg oder Niederlage. Aus diesem Grunde untersuchte die CPRG die Möglichkeit, analog zur Minimax-Suche für Spiele mit vollständiger Information oder aber der Expectimax-Suche für Spiele mit vollständiger Information und stochastischem Element, einen Algorithmus für Spielbäume mit unvollständiger Information und stochastischen Ereignissen zu entwickeln. Für Spiele perfekter Information steht schon seit langem der sehr erfolgreich eingesetzte *Minimax-Algorithmus* mit zahlreichen Verbesserungen wie z.B. *α/β -Pruning* zur Verfügung [RN10]. Handelt es sich dagegen um ein Spiel wie Backgammon, bei dem stochastische Ereignisse, wie zum Beispiel das Werfen mehrerer Würfel, hinzutreten, leistet beispielsweise der *Expectimax-Algorithmus* gute Arbeit, der in [Ven06] erläutert wird. Für das Texas Hold'em hingegen können beide Verfahren nicht eingesetzt werden, da die Knoten des zugehörigen Spielbaums nicht unabhängig voneinander sind, sondern teilweise zur gleichen

Informationsmenge gehören. Alle Knoten im Spielbaum, in denen sich ein Spieler aufgrund der unbekannten gegnerischen Karten *möglicherweise* befindet, gehören zu einer Informationsmenge, deshalb muss das Vorgehen für alle diese Knoten das gleiche sein. Abbildung 2.1 verdeutlicht diesen Sachverhalt.

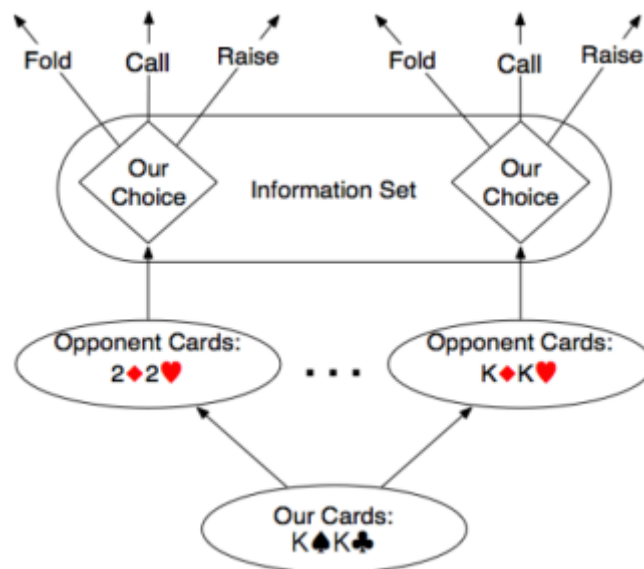


Abbildung 2.1: Informationsmenge im Texas Hold'em Spielbaum (entnommen aus [Joh07]). Alle möglichen gegnerischen Hände gehören zur selben Informationsmenge. Der Spielbaum lässt sich verkleinern, indem eine Wahrscheinlichkeitsverteilung über die gegnerischen Hände angenommen wird.

Es ist möglich, alle gegnerischen Entscheidungsknoten innerhalb einer Informationsmenge zu einem Knoten zusammenzufassen und eine Wahrscheinlichkeitsverteilung über alle möglichen gegnerischen Hände anzunehmen. Die gegnerischen Hände sind, wie bereits bemerkt, nicht gleichverteilt, sondern schwache Hände werden eher „weggeworfen“ als starke Hände, die eventuell bis zu einem Showdown gespielt werden. Die entscheidenden Punkte für alle Berechnungen auf dem Spielbaum sind eine möglichst gute Schätzung der Stärke der gegnerischen Hand und die Wahrscheinlichkeiten der gegnerischen Aktionen, die dann dazu benutzt werden, den Erwartungswert der drei uns zur Verfügung stehenden Aktionen *aussteigen*, *schieben/mitgehen*, *setzen/erhöhen* an den eigenen Entscheidungsknoten zu berechnen. Die Blattknoten des Spielbaumes stellen das Ende des

Spiels dar. Im einfachen Fall, wenn ein Spieler aussteigt – dann ist das Ermitteln des Siegers trivial – oder aber, wenn es zu einem Showdown kommt. Hier errechnet dann eine Heuristik die Gewinnwahrscheinlichkeit und gibt den errechneten Erwartungswert weiter nach oben im Baum. Für den Fall, dass wir uns immer dazu entscheiden, an den eigenen Entscheidungsknoten (siehe unten) den höchsten Erwartungswert aller möglichen Aktionen weiter nach oben im Baum zu propagieren, heißt der Algorithmus *Minimax*. Entscheiden wir uns aber dafür, da dieses Vorgehen möglicherweise Rückschlüsse über die Spielweise liefern kann, für die eigenen Knoten eine gemischte Strategie anzuwenden, so wird dieser Algorithmus *Minimax* genannt. Der Suchbaum besteht aus vier Arten von Knoten, die sich hinsichtlich ihrer Eigenschaften unterscheiden:

- **Gegnerische Entscheidungsknoten**

Sei $W(G_i)$ die Wahrscheinlichkeit¹⁶ jedes Zweiges i (es existieren drei Zweige, die zu Kindknoten führen: aussteigen, schieben/mitgehen, setzen/erhöhen) an einem gegnerischen Entscheidungsknoten G . Dann ist der Erwartungswert des Knotens G die gewichtete Summe:

$$EW(G) = \sum_{i \in \{a, m, e\}} W(G_i) \times EW(G_i) \quad (2.2)$$

- **Eigene Entscheidungsknoten**

An eigenen Entscheidungsknoten E können wir entweder eine gemischte Strategie anwenden, oder uns für den maximalen Erwartungswert aller Aktionen entscheiden. In diesem Falle gilt für die drei existierenden Zweige, die den zur Verfügung stehenden Aktionen entsprechen:

$$EW(E) = \max(EW(E_a), EW(E_m), EW(E_e)) \quad (2.3)$$

¹⁶ Die Wahrscheinlichkeiten für die Zweige, d.h. für die gegnerischen Aktionen liefert das Gegnermodell.

- **Blattknoten**

Sei B ein Blattknoten, W_{Sieg} die Wahrscheinlichkeit den Pot zu gewinnen, $Potsize$ die Größe des Pots am betrachteten Knoten und $Kosten$ die Kosten, um den Blattknoten zu erreichen (für einen Showdown betragen diese $0,5 \times Potsize$). Falls der Blattknoten durch ein Aussteigen eines Spielers aus dem Spiel erreicht wurde, so ist W_{Sieg} entweder 1 oder 0. Der Erwartungswert für einen Blattknoten errechnet sich folgendermaßen:

$$EW(B) = (W_{Sieg} \times Potsize) - Kosten \quad (2.4)$$

- **Zufallsknoten**

Der Erwartungswert der Zufallsknoten im Spielbaum ist die gewichtete Summe der Erwartungswerte aller Teilbäume, die zu allen möglichen Zufallsereignissen, d.h. den möglichen Gemeinschaftskarten (3 auf dem Flop und jeweils eine Karte am Turn und River), die am Knoten ausgeteilt werden, gehören. Sei $W(Z_i)$ die Wahrscheinlichkeit jedes Zweiges i am Zufallsknoten Z , und sei n die Anzahl der Zweige an Z . Dann ist der Erwartungswert des Knotens Z :

$$EW(Z) = \sum_{1 \leq i \leq n} W(Z_i) \times EW(Z_i) \quad (2.5)$$

2.1.5.3.1 BERECHNUNG DES ERWARTUNGSWERTES

Für jeden Blattknoten, an dem ein Showdown stattfindet, wird ein Histogramm der Handstärke¹⁷ des Gegners bereit gehalten, das sich in 20 Intervalle gliedert, die alle die gleiche Breite von 0,05 besitzen. Diese Histogramme werden benutzt, um auf die ungefähre Handstärke des Gegners für eine gegebene Setz-Sequenz zu schließen. Nach jedem Spiel, für das ein Showdown beobachtet wurde, werden sie aktualisiert. Betrachtet werden in den Situationen, in denen eine Annahme über die Handstärke des Gegners getroffen werden muss, immer die Histogramme, die zu einer exakt gleichen Setz-Sequenz gehören.

Seien wir beispielsweise SP1, der Gegner SP2, im Pot befinden sich 4 Small Bets und es ist die letzte Setzrunde. Den zugehörigen Baum, der das Setzverhalten beschreibt, zeigt Abbildung 2.2¹⁸. Eigene Entscheidungsknoten sind als Kreis dargestellt, während die gegnerischen Entscheidungsknoten durch Quadrate mit abgerundeten Ecken symbolisiert werden. Spielendknoten, die hier als Sechsecke repräsentiert werden, tragen für den Fall, dass sie durch einen Fold erreicht wurden, den bekannten Erwartungswert als Beschriftung. Showdown-Knoten, für die der Erwartungswert geschätzt werden soll, sind mit einem Fragezeichen beschriftet. Darüber hinaus sind in Abbildung 2.2 zwei Histogramme dargestellt. Die Beschriftung $P2[bRrC]$ bedeutet hier, dass es sich um das Histogramm für SP2 für die Setz-Sequenz $bRrC$ handelt.

¹⁷ Die Handstärke bezeichnet die Wahrscheinlichkeit, dass eine gegebene Hand besser ist als die Hand des Gegenspielers. Zur Berechnung werden alle möglichen gegnerischen Hände betrachtet und mit der eigenen verglichen.

¹⁸ Zu beachten ist hier, dass das Beispiel aus [BDSB04] entnommen wurde und die Notation für die Aktionen von der in Abschnitt 2.1.4 eingeführten Notation abweicht. Die Aktionen unseres Gegners sind hier in Großbuchstaben notiert, während unsere eigenen Aktionen in Kleinbuchstaben dargestellt sind.

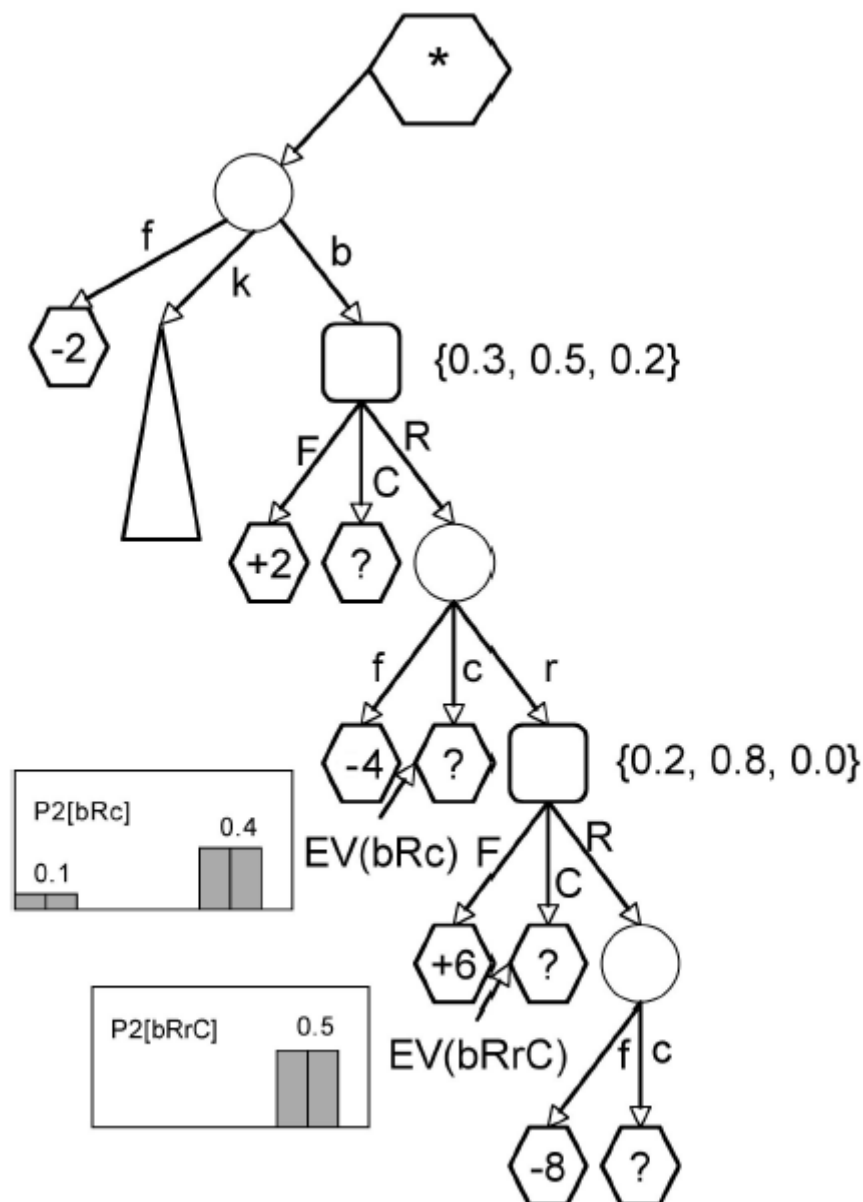


Abbildung 2.2: Setz-Baum für die Berechnung des Erwartungswertes am eigenen Entscheidungsknoten (entnommen aus [BDSB04] - ein Fehler wurde behoben). Eigene Entscheidungsknoten sind als Kreise dargestellt. An gegnerischen Entscheidungsknoten, die durch Quadrate mit abgerundeten Ecken symbolisiert werden, liefert das Gegnermodell Wahrscheinlichkeiten für die drei zur Verfügung stehenden Aktionen des Gegners.

Nach einem Einsatz von 2 Small Bets unsererseits, erhöht der Gegner um weitere 2 SB. Interessant für uns ist nun eine mögliche Wahrscheinlichkeitsverteilung der gegnerischen Handstärken. Zur Vereinfachung werden im Folgenden Histogramme mit 10 Intervallen betrachtet. Nehmen wir an, wir hätten für die exakte, uns vorliegende Setz-Sequenz ein Histogramm [1 1 0 0 0 0 0 4 4 0]. Dieses Histogramm zeigt uns, dass in vergangenen Spielen

die Erhöhung des Gegners zu 20% ein Bluff war und zu 80% auf eine Handstärke zwischen 0,7 und 0,9 schließen lässt. Nehmen wir weiterhin an, wir erhöhen nun unsererseits und der Gegner geht mit (*bRrC*). Wieder steht uns ein Histogramm zur Verfügung: [0 0 0 0 0 0 5 5 0]. Daraus lässt sich ein Wahrscheinlichkeitstripel ableiten: $W(\text{aussteigen}, \text{mitgehen}, \text{erhöhen}) = (0.2, 0.8, 0.0)$. Um unsere nächste Aktion zu planen, berechnen wir den EW für alle uns zur Verfügung stehenden Aktionen: $EW(\text{aussteigen})$, $EW(\text{mitgehen})$ und $EW(\text{erhöhen})$. Die Berechnung von $EW(\text{aussteigen})$ ist trivial – das Spiel hat uns 4 SB gekostet.

$EW(\text{mitgehen})$ hängt direkt von unserer Handstärke ab, da diese unsere Chance auf einen Sieg ausdrückt. Eine Handstärke zwischen 0,2 und 0,7 reicht nur aus, um einen Bluff zu schlagen, also ist $W(\text{sieg} | bRc) = 0,2$. Da der Pot am Ende eine Größe von 12 SB hat, zu dem wir 6 SB beigesteuert haben, ist $EW(\text{mitgehen}) = -3,6$. Da $EW(\text{mitgehen}) > EW(\text{aussteigen})$, werden wir für eine Handstärke zwischen 0,2 und 0,7 nicht aus dem Spiel aussteigen. Um den EW für ein erneutes Erhöhen unsererseits zu berechnen, müssen wir alle Unterbäume betrachten: *bRrF*, *bRrC*, *bRrFf* und *bRrRc*. Da $W(\text{erhöhen})$ des Gegners null ist, können die letzten beiden Fälle ignoriert werden. Der EW von *bRrF* beträgt $0,2 \times 6 = +1,2$ SB und hängt nicht von unserer Handstärke ab. Der EW für *bRrC* hängt wieder vom zugehörigen Histogramm ab, in diesem Falle gilt: $EW(\text{erhöhen}) = 1,2 + 0,8 \times (16 \times W(\text{sieg} | bRrC) - 8)$.

Daraus folgt, dass wir im Falle einer maximierenden Strategie jede Hand wegwerfen würden, deren Handstärke kleiner als 0,167 wäre. Hingegen würden wir für eine Handstärke zwischen 0,167 und 0,8 mitgehen und ab einer Handstärke von 0,8 erhöhen.

Die Vorgehensweise zur Ermittlung der gegnerischen Aktionen und der Gewinnwahrscheinlichkeiten ist in der Praxis etwas komplexer als oben beschrieben. Beispielsweise existieren 13.122 Setz-Sequenzen, die zu einem Showdown führen können. Das bedeutet, dass das Gegnermodell sehr langsam lernen würde. Aus diesem Grunde existieren weitere Histogramme der gegnerischen Handstärken, die vom Spielkontext abstrahieren, beispielsweise für die Größe des Pots oder aber das Aggressionsverhalten des Gegners. Die verschiedenen Histogramme, die bei der Baumsuche dann zur Verfügung stehen, werden abhängig von der Zahl der Beobachtungen für die jeweiligen Histogramme gewichtet.

Der zentrale Punkt des von der CPRG für die adaptive Spielbaumsuche eingesetzten Gegnermodells ist, dass es sich um ein von Hand erstelltes und parametrisiertes Modell handelt. Ein detaillierterer Überblick über den genauen Aufbau findet sich in [BDSB04], zusammen mit der Bemerkung, dass dieses Modell noch viel Raum für Verbesserungen bietet.

2.1.6 Kombination verschiedener Ansätze

Sowohl der spieltheoretische Ansatz als auch die adaptive Spielbaumsuche besitzen Vor- und Nachteile, die sich durch eine Kombination beider Verfahren zumindest abmildern lassen. An dieser Stelle soll nur in aller Kürze die prinzipielle Vorgehensweise erläutert werden.

Eine einfache Möglichkeit zur Kombination beider Vorgehensweisen ist beispielsweise, beide Verfahren zu implementieren und dann je nach Situation die passende Komponente spielen zu lassen. In der Praxis sähe das so aus, dass zu Beginn einer Serie von Spielen, wenn noch wenige Informationen über die Spielweise des Gegners zur Verfügung stehen, zunächst die spieltheoretische Komponente zum Einsatz kommt, d.h. spielt. Wie oben beschrieben ist dies ein defensiver Ansatz. Es werden im Grunde keine Fehler gemacht, suboptimales Spielen des Gegners wird allerdings auf lange Sicht bestraft - möglicherweise aber nur in sehr kleinem Umfang.

Nach einer gewissen Menge an Spielen ist das Gegnermodell, das in der adaptiven Spielbaumsuche zum Einsatz kommt, in der Lage den Gegner relativ genau zu modellieren. Nun kann die adaptive Komponente in bestimmten Situationen, in denen das Gegnermodell hinreichend sicher ist, gute Ergebnisse zu liefern, zum Einsatz gebracht werden, um offensiv die Schwächen des Gegners auszunutzen. Problematisch hierbei ist die Abschätzung, wann das Gegnermodell in der Lage ist, adäquate Ergebnisse zu liefern. In Abschnitt 4.2 wird versucht, eine Antwort auf diese Frage zu finden.

2.2 KLASSIFIKATION

Da der Lösungsansatz, der in dieser Arbeit entwickelt werden soll, als Klassifikationsproblem formuliert ist, erfolgt an dieser Stelle eine kurze Darstellung der Prinzipien der Klassifikation. Eine ausführliche Einführung in die Thematik und in die verschiedenen Verfahren bieten insbesondere [HK06], [WFH11] und [RN10].

Formal betrachtet ist ein Klassifikator eine Funktion, die einem Merkmalsvektor eine Klasse¹⁹ zuordnet. Ein Merkmalsvektor ist eine geordnete Menge numerischer oder kategorischer Merkmale bzw. Attribute, die eine bestimmte Beobachtung, Situation oder Messung charakterisiert. Die zuzuordnende bzw. vorauszusagende Klasse ist dabei kategorischer²⁰ Natur.

Klassifikation ist im Allgemeinen ein zweistufiger Prozess. Im ersten Schritt wird der Klassifikator aufgebaut. Dies geschieht anhand eines Trainingsdatensatzes bzw. einer Trainingsmenge. Diese ist dadurch charakterisiert, dass für die in ihr enthaltenen Merkmalsvektoren – auch Beispiele, Beobachtungen oder Instanzen genannt – ein ausgezeichnetes Merkmal existiert, das die Klassenzugehörigkeit beschreibt, im Folgenden *Label* oder *Markierung* genannt. Im zweiten Schritt kann ein Klassifikator dann dazu benutzt werden, bisher unbekannten Merkmalsvektoren eine Klassenzugehörigkeit zuzuordnen.

Soll keine Klassenzugehörigkeit vorausgesagt werden, sondern ein numerischer Wert, so spricht man von *Regression*. Da bereits bekannte Daten zum Aufbau der Klassifikations- bzw. Regressionsmodelle herangezogen werden, werden diese zum *überwachten Lernen* gezählt. Im Gegensatz dazu wird beispielsweise das *Clustering*, bei dem a priori keine Informationen über eine Klassenzugehörigkeit bekannt sind, dem *unüberwachten Lernen* zugeordnet.

Ein allgemeines Problem, das bei der Klassifikation auftritt, ist die ungleichmäßige Verteilung der vorauszusagenden Klassen. Der ideale Fall, dass die relativen Häufigkeiten der Klassen gleichverteilt sind, tritt nur selten ein. Dies hat im Allgemeinen negative Auswirkungen – insbesondere auf die Vorhersagegenauigkeit der unterrepräsentierten Klassen. In [CBHK02]

¹⁹ Die *Multilabel-Klassifikation*, bei der ein Merkmalsvektor mehreren Klassen zugeordnet wird, soll hier nicht näher betrachtet werden. Eine Einführung hierzu liefert [TK07].

²⁰ Dies bedeutet insbesondere, dass die Klasseneinteilung ungeordnet und diskret ist.

und [WP01] werden diese Effekte beschrieben und Lösungsansätze skizziert, die diese Problematik zumindest abschwächen sollen.

Für die Klassifikation existiert heute eine Vielzahl verschiedener Algorithmen, die sich für eine gegebene Aufgabenstellung mehr oder weniger gut eignen. Bekannte grundlegende Klassifikationsverfahren sind beispielsweise *Support-Vektor-Maschinen*, *künstliche neuronale Netze (KNN)*, *Entscheidungsbäume* und *Bayessche Netze*.

Für jeden Klassifikationsalgorithmus existiert ein Datensatz, auf dem er sehr genau arbeitet, und ein anderer Datensatz, auf dem er eine nur mangelhafte Klassifikationsleistung erbringt [Alp08]. Insbesondere ist zu erwähnen, dass es keinen „besten“ Klassifikationsalgorithmus gibt, der für alle Probleme die besten Ergebnisse liefert. Dies spiegelt sich in dem berühmten *No-Free-Lunch-Theorem (Nichts-ist-umsonst-Theorem)* von Wolpert wider [Wol95].

Im Folgenden wird das Vorgehen beim Aufbau und Einsatz von Entscheidungsbäumen betrachtet, zu denen der in dieser Arbeit verwendete CART-Algorithmus gezählt wird.

2.2.1 Entscheidungsbäume

Entscheidungsbäume sind eine Familie nichtparametrischer Methoden des überwachten Lernens, die sowohl für die Regression als auch zur Klassifikation eingesetzt werden können. Zwar existieren seit den frühen 60er Jahren einfache Entscheidungsbaum-Varianten, z.B. *AID (Automatic Interaction Detection)*, das zur Regression eingesetzt wurde [SM64], doch erfreuten sich Entscheidungsbäume erst durch die Arbeiten von Quinlan [Qui86] und Breiman et al. [BFOS84] einer größeren Akzeptanz und Verbreitung. Im Folgenden werden binäre²¹, univariate²² Bäume betrachtet, die zur Klassifikation eingesetzt werden.

Entscheidungsbäume sind geordnete und gerichtete Bäume, die aus einem Wurzelknoten, einer Menge innerer Entscheidungsknoten, und mindestens zwei Blattknoten bestehen.

²¹ Binär bedeutet, dass jeder Knoten, der kein Blattknoten ist, genau zwei Kindknoten besitzt.

²² Bei univariaten Bäumen wird an jedem Entscheidungsknoten nur eine der Eingabedimensionen für den Test verwendet.

DIE KONSTRUKTION VON ENTSCHEIDUNGSBÄUMEN

Für eine gegebene Trainingsmenge M mit n Trainingsdaten und Z Klassen beginnt die Konstruktion eines Entscheidungsbaumes an der Wurzel. Die gesamte Trainingsmenge soll anhand eines Attribut/Wert-Paares²³ möglichst optimal in zwei disjunkte Teilmengen aufgespalten werden. Die Güte der Aufteilung bzw. des *Splits* lässt sich unter Zuhilfenahme eines *Verunreinigungsmaßes* ermitteln. Häufig, beispielsweise bei dem von Quinlan entwickelten *ID3-Algorithmus* [Qui86], wird die *Entropie* zur Quantifizierung der Verunreinigung verwendet:

$$H_M = - \sum_{i=1}^Z p_i \log_2 p_i \quad (2.6)$$

Dabei bezeichnet p_i die relative Häufigkeit der Klasse C_i in der Trainingsmenge. Ein weiteres Maß der Unreinheit, das häufig - insbesondere auch bei CART - Verwendung findet, ist der *Gini-Index*, der als

$$\text{gini}_M = 1 - \sum_{i=1}^Z p_i^2 \quad (2.7)$$

definiert ist. Um die Qualität einer Aufteilung zu ermitteln, wird die Differenz aus der Verunreinigung vor der Aufteilung und der gewichteten Summe der Verunreinigungen der beiden Teilmengen nach der Aufteilung berechnet. Weitergehende Untersuchungen [RS04] zeigen, dass sich die Splits, die durch den Gini-Index erzeugt werden, in weniger als 2% aller Fälle von den Splits unterscheiden, die durch eine Abnahme der Entropie erzeugt wurden.

Rekursiv werden nun die Teilmengen, die sich durch diese lokal optimalen Aufteilungen ergeben, weiter geteilt, bis ein *Stopp-Kriterium* erfüllt und ein Blattknoten im zu erstellenden Baum erreicht ist. Dies kann beispielsweise eine bestimmte Anzahl der verbleibenden Trainingsbeispiele sein, oder aber, wenn eine verbleibende Teilmenge nur noch Instanzen

²³ Dieses Attribut/Wert-Paar entspricht dann bei der Klassifikation einer Instanz der Eingabedimension i , bzw. dem Schwellenwert ω_m .

einer Klasse enthält. Das Blatt trägt das Label der Klasse, die in der verbleibenden Trainingsmenge am häufigsten auftritt. Zusätzlich werden meist auch die absoluten und/oder relativen Häufigkeiten aller verbleibenden Klassen gespeichert, um diese dann bei Bedarf bereitzustellen.

PRUNING

Eine wichtige Anforderung an Klassifikationsverfahren ist die Fähigkeit zur *Generalisierung*, d.h. die korrekte Klassifizierung bisher unbekannter Instanzen. Werden Entscheidungsbäume vollständig aufgebaut, sodass alle Blätter nur noch Instanzen einer Klasse beinhalten, oder die Verunreinigung sich nicht mehr reduzieren lässt, dann beruhen die Klassifikationsentscheidungen in der Regel auf einer oder sehr wenigen Instanzen, wodurch die Generalisierungsfähigkeit durch eine hohe Varianz stark negativ beeinflusst wird [Alp08]. Eine derartige *Überanpassung* an die Trainingsdaten lässt sich durch das sogenannte *Pruning* vermeiden.

Bei der Variante des *Prepruning* wird die Konstruktion des Baumes abgebrochen, noch bevor dieser vollständig aufgebaut ist. Dies geschieht in der Regel durch die Spezifikation einer minimalen Anzahl an Instanzen, die ein Blattknoten beinhalten muss, oder aber durch das Unterschreiten eines Schwellenwertes für die Abnahme der Verunreinigung.

Eine aufwendigere Methode, die jedoch in der Praxis im Allgemeinen zu einer besseren Generalisierungsfähigkeit der Entscheidungsbäume führt, ist das *Postpruning*. Hier wird zunächst der vollständige Baum aufgebaut, und dann werden iterativ die Unterbäume entfernt, die eine Überanpassung verursachen. Bekannte Pruning-Verfahren sind u. a. *Reduced-Error Pruning*, *Pessimistic-Error Pruning* und *Minimal Cost-Complexity Pruning (MCCP)*, das auch bei CART verwendet wird.

Der Grundgedanke des MCCP ist es, einen möglichst guten Kompromiss zwischen den „Kosten“ des beschnittenen Baumes, die von der Anzahl der Blätter abhängig sind, und der Klassifikationsleistung zu finden. Im Folgenden wird in aller Kürze das prinzipielle Vorgehen des MCCP beschrieben. Die genaue Arbeitsweise und insbesondere die Beweise sind äußerst komplex, daher sei an dieser Stelle auf eine detailliertere Darstellung in [BFOS84] verwiesen.

Sei $|T|$ die Anzahl der Blätter des Baumes T und $R(T)$ der Resubstitutionsfehler, d.h. die Missklassifikationsrate²⁴ von T für der Trainingsmenge, anhand derer der Baum erstellt wurde. Dann findet das MCCP den Teilbaum von T , für den die *Kosten-Komplexität (cost complexity)* $R_\alpha(T)$ minimal ist:

$$R_\alpha(T) = R(T) + \alpha|T| \quad (2.8)$$

Dabei ist $\alpha \geq 0$ eine reelle Zahl, der sogenannte *Komplexitätsparameter*, und beschreibt die Kosten für jedes Blatt. Es lässt sich zeigen, dass für jeden Wert von α ein eindeutiger Teilbaum von T existiert, der $R_\alpha(T)$ minimiert.

Für sehr kleine Werte von α ist die Kosten-Komplexität fast ausschließlich durch den Resubstitutionsfehler bestimmt, und der vollständige Baum wäre optimal. Für sehr große Werte von α würde ein Baum, der nur aus der Wurzel besteht, als der beste angesehen. Um den optimalen Baum zu finden, der $R_\alpha(T)$ für alle α minimiert, wird das *Weakest Link Cutting* eingesetzt. Dabei werden iterativ Teilbäume abgeschnitten²⁵, sodass möglichst viele Blätter entfernt werden, der Resubstitutionsfehler der so beschnittenen Bäume aber möglichst wenig steigt. Da die Abschätzung der Klassifikationsleistung durch den Resubstitutionsfehler übermäßig optimistisch ist, wird in der Praxis meist eine interne K -fache Kreuzvalidierung²⁶ (siehe 2.2.2.1) verwendet, um die Klassifikationsleistung der beschnittenen Bäume zu bewerten.

²⁴ Die Missklassifikationsrate ist der Anteil der falsch klassifizierten Instanzen an der Gesamtheit aller klassifizierten Instanzen und entspricht $1 - KKR$ (siehe 2.2.2.2).

²⁵ Das bedeutet, dass alle Instanzen, die sich in den Blättern des jeweiligen abgeschnittenen Teilbaumes befinden, dem Knoten zugeordnet werden, in dem der Baum beschnitten wird. Im Anschluss daran, wird dieser Knoten mit dem entsprechenden Label der Klasse mit den meisten Instanzen in diesem Knoten versehen.

²⁶ Für diese Arbeit wurde CART in der von WEKA (siehe 3.4.1) bereitgestellten Implementierung *SimpleCart* verwendet, die auch in *RapidMiner* (siehe 3.4.2) verfügbar ist. Die hier verwendete Parametrisierung von CART entspricht den voreingestellten Standardwerten in Weka/RapidMiner. Insbesondere kommt eine 5-fache interne Kreuzvalidierung zum Einsatz. Die genaue Parametrisierung findet sich auf der beiliegenden CD-ROM.

Breiman et al. zeigen in [BFOS84], dass sich die Wahl der Pruning-Methode weitaus stärker auf die Klassifikationsgüte auswirkt als das verwendete Verunreinigungsmaß.

KLASSIFIKATION DURCH ENTSCHEIDUNGSBÄUME

Um eine vorliegende Instanz zu klassifizieren, wird - beginnend an der Wurzel des Baumes - an jedem Entscheidungsknoten ein Test durchgeführt, der entscheidet, welcher Pfad in Richtung der Blattknoten weiter verfolgt wird. Für numerische Attributwerte stellt jeder Test an einem Entscheidungsknoten m eine *Diskriminante* $f_m(x)$ dar, die dem Vergleich des Attributwerts der i -ten Eingabedimension des Eingabevektors x mit einem Schwellenwert ω_m entspricht.

Für nominale Attribute wird i. Allg. auf die Gleichheit eines Attributwertes mit einem nominalen Testwert oder die Zugehörigkeit zu einer Menge getestet. Die Blätter enthalten jeweils das Label der Klasse, der die Instanz, die das Blatt durch wiederholte Tests in den Entscheidungsknoten erreicht hat, zugeordnet wird. Oftmals stellen Entscheidungsbäume an den Blattknoten auch Wahrscheinlichkeiten zur Verfügung, um abzuschätzen, wie wahrscheinlich eine Instanz einer bestimmten Klasse, bzw. den anderen möglichen Klassen angehört. Diese werden beim Aufbau des Baumes gewonnen (siehe oben).

Abbildung 2.3 zeigt einen sehr einfachen Klassifikationsbaum, so wie er auch von CART aufgebaut wird, der zur Bestimmung der Wahrscheinlichkeiten der möglichen gegnerischen Aktionen eingesetzt werden könnte. Die Knoten des Baumes, an denen ein Test für ein Attribut durchgeführt wird, sind schwarz umrandet und tragen als Beschriftung den Test, der in diesem Knoten durchgeführt wird.

Nehmen wir an, wir wollen ermitteln, welche Aktion ein Gegner durchführt, nachdem wir einen Einsatz bzw. Bet tätigen. Seien zwei der Merkmale beispielsweise die Größe des Pots (*Potsize*) und die Beobachtung, ob sich mit den gegebenen und zukünftigen Gemeinschaftskarten unter Beachtung der Hole Cards bis zum Ende des Spiels ein Flush bilden lässt oder nicht²⁷.

²⁷ Um einen Flush zu bilden, sind fünf Karten der gleichen Farbe (Pik, Kreuz, Karo oder Herz) erforderlich. Das bedeutet, dass auf dem Flop immer die Möglichkeit besteht, dass bis zur letzten Spielrunde ein Flush gebildet

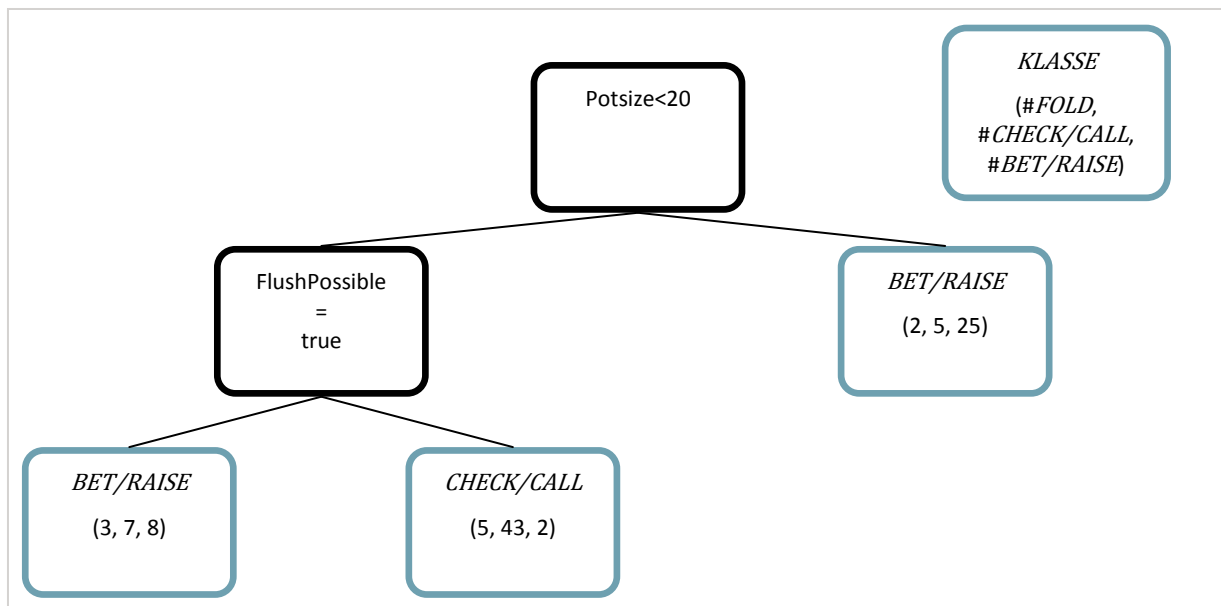


Abbildung 2.3: Stark vereinfachtes Beispiel eines Klassifikationsbaums zur Ermittlung der Wahrscheinlichkeiten für die möglichen gegnerischen Aktionen. Durch wiederholte Tests in den schwarz umrandeten Knoten wird ein Blatt erreicht, das die vorliegende Instanz einer bestimmten Klasse zuordnet. Zusätzlich lassen sich an den Blättern Wahrscheinlichkeiten für die Zugehörigkeit zu den Klassen ermitteln.

Der erste Test findet im Wurzelknoten statt. Hier wird beispielsweise geprüft, ob die Potgröße kleiner als 20 ist (*Potsize<20*). Im negativen Fall, d.h. bei einer Potgröße von mindestens 20 wird hier der rechte Zweig gewählt und das so erreichte Blatt würde die vorliegende Instanz als der Klasse *BET/RAISE* zugehörig und somit als eine Erhöhung klassifizieren. Falls der Test positiv ausfällt, wird der linke Zweig gewählt. Nun wird geprüft, ob das Merkmal *FlushPossible* für die vorliegende Instanz den Wert *true* trägt. Wäre dies hier nicht der Fall, dann würde der rechte Zweig gewählt, und das so erreichte Blatt würde die Instanz als *CHECK/CALL*, d.h. in diesem Falle als ein Mitgehen klassifizieren. Dieses Blatt umfasst 50 Trainingsbeispiele, von denen fünf der Klasse *FOLD*, zwei der Klasse *BET/RAISE* und 43 der Klasse *CHECK/CALL* angehören. Das bedeutet, dass dieses Blatt die Wahrscheinlichkeiten von $\frac{5}{50} = 0,1$ für ein Aussteigen, $\frac{2}{50} = 0,04$ für eine Erhöhung und $\frac{43}{50} = 0,86$ für ein Mitgehen des Gegners liefern würde.

werden kann. Für den Turn müssen mindestens zwei der Gemeinschaftskarten die gleiche Farbe besitzen - auf dem River mindestens drei.

CART

Der von Breiman et al. Ende der 70er und zu Beginn der 80er Jahre entwickelte CART-Algorithmus war, wie bereits erwähnt, nicht das erste Entscheidungsbaum-Verfahren. Dennoch wurde CART weitgehend ignoriert und Breiman et al. war es nicht möglich, ihr Verfahren in angesehenen Publikationen der Statistik-Gemeinde zu publizieren. Aus diesem Grunde veröffentlichten Breiman et al. 1984 die „CART-Monographie“ [BFOS84].

Wie oben bereits beschrieben, wird von CART ein univariater, binärer Entscheidungsbaum aufgebaut. Als Verunreinigungsmaß wird der Gini-Index verwendet und nach dem Aufbau des Baumes kommt das Minimal Cost-Complexity Pruning zum Einsatz.

Die Möglichkeit, Entscheidungsbäume zur Modellierung gegnerischer Aktionen beim Texas Hold'em zu verwenden, wurde bereits in [Dav02] erwähnt. Der CART-Algorithmus verfügt über einige interessante Vor- und Nachteile, die sich beispielsweise in [Tim04] finden. Im Folgenden werden diese kurz vorgestellt und im Hinblick auf den Einsatz in einem Modell zur Vorhersage der gegnerischen Aktionen kurz kommentiert.

Vorteile von CART

V1: Eine Einschränkung/Selektion der Eingabevariablen ist nicht notwendig. CART identifiziert die relevanten Eingabevariablen. Nicht relevante Variablen werden beim Aufbau des Baumes i. Allg. nicht beachtet.

V2: CART ist interpretierbar. Im Gegensatz zu etwa einem KNN, ist das Ergebnis des CART-Algorithmus intuitiv verständlich. Darüber hinaus lassen sich die von CART aufgebauten Bäume leicht in konjunktive Regeln umwandeln.

V3: Invarianz bezüglich monotoner Transformation der Variablen. Durch eine monotone Transformation einer oder mehrerer Eingabevariablen ändert sich der resultierende Baum nicht. Lediglich die Schwellenwerte in den inneren Knoten ändern sich.

V4: CART ist robust gegenüber Ausreißern. Viele Verfahren des maschinellen Lernens sind anfällig gegenüber Ausreißern - zum Beispiel hervorgerufen durch Messfehler. CART ist robust gegenüber Ausreißern, und es ist keine weitere Vorverarbeitung der Eingabevektoren erforderlich, um diese zu eliminieren.

V5: CART kann mit stetigen und diskreten Eingabevariablen umgehen.

V6: CART verfügt über Techniken zum Umgang mit fehlenden Eingabevariablen.

Zur Konstruktion eines Gegnermodells scheint der CART-Algorithmus ein idealer Kandidat zu sein. Vorteil V1 eröffnet beispielsweise die Möglichkeit, Merkmalsvektoren mit relativ vielen *plausiblen Attributen* zu erzeugen und diese effektiv zu testen und einzusetzen. Eine aufwendige Selektion von Attributen - beispielsweise durch eine *Hauptkomponentenanalyse (PCA)*, siehe [Alp08]) - entfällt. Auch Vorteile V3, V5 und V6 sind in ähnlicher Weise zu deuten, d.h. CART ist bezüglich der Eingaben ein relativ robuster Algorithmus und erfordert i. Allg. keine Vorverarbeitung der Eingaben.

Besonders interessant für die Gegnermodellierung ist auch Vorteil V2. Die von CART erzeugten Bäume sind intuitiv verständlich bzw. interpretierbar und die Klassifikation anhand dieser Bäume erfolgt durch iterative Tests. Dieses schrittweise Vorgehen ist der menschlichen Entscheidungsfindung in vielen Situationen sehr ähnlich, z.B. bei einer medizinischen Diagnose durch einen Arzt. Es lässt sich argumentieren, dass die Art und Weise, in der Pokerspieler ihre Entscheidungen treffen, mit diesem Vorgehen weitgehend identisch ist. Möglicherweise lassen sich durch diese Betrachtungsweise Attribute finden, mit denen CART besonders gute Ergebnisse erzielen kann.

Ein wichtiger Bestandteil des Pokerspiels ist die Täuschung der Gegenspieler. Von Neumann und Morgenstern demonstrierten bereits 1944 in [NM44] anhand einer vereinfachten Pokervariante, dass Bluffing wesentlicher Bestandteil einer optimalen Strategie ist. Bluffing ist zwar nicht die einzige Möglichkeit, den Gegenspieler zu täuschen, doch verdeutlicht dies insbesondere, dass Spieler in zwei exakt gleichen Situationen verschiedene Aktionen tätigen werden. Dies ist nur einer der Faktoren, die zu Ausreißern in den Trainingsdaten für ein Gegnermodell führen können. Vorteil V4 zeigt, dass CART diese Anforderung erfüllt.

Diesen Vorteilen stehen einige Nachteile gegenüber:

Nachteile von CART

N1:*Instabile Bäume*. Das Entfernen oder Hinzufügen von nur wenigen Trainingsdaten kann eine radikale Änderung in der Struktur und Größe des resultierenden Klassifikationsbaumes zur Folge haben.

N2:*Splits erfolgen eindimensional.* Jede Aufteilung der Trainingsdaten in einem inneren Knoten erfolgt lediglich durch Vergleich *einer* Variablen mit *einem* Schwellenwert.

N3:Es existieren heute viele neuere Verfahren, die i.d.R. *bessere Ergebnisse liefern als CART.*

Nachteil N1 konnte zwar in den Versuchen dieser Arbeit beobachtet werden, jedoch scheint sich dieses Verhalten nicht nachteilig auf die Klassifikationsleistung auszuwirken (siehe 4.2.1). Die Tatsache, dass Splits nur eindimensional erfolgen, kann auch als Vorteil gewertet werden und korrespondiert mit Vorteil V2, d.h. das Vorgehen bei der Klassifikation entspricht einer menschlichen Entscheidungsfindung.

Der letzte Nachteil konnte für die hier gewählte Repräsentation der Merkmalsvektoren in Vorab-Tests nicht bestätigt werden, bietet jedoch interessante Ansätze für weitere Arbeiten.

2.2.2 Bewertung von Klassifikatoren

In der Praxis wird eine Menge von Merkmalsvektoren, für die eine Klassenzugehörigkeit bekannt ist, üblicherweise in zwei disjunkte Untermengen aufgeteilt. Zum Einen in eine Trainingsmenge, anhand derer der Klassifikator aufgebaut wird. Zum Anderen in eine Testmenge, die dazu dient, die Leistung des Klassifikators auf bisher unbekannten Daten abzuschätzen. Üblicherweise werden $\frac{2}{3}$ der *gelabelten* Vektoren als Trainingsdatensatz verwendet und der Rest zur Evaluierung der Performanz [HK06]. Dieses Aufteilen in eine Trainings- und eine Testmenge ist jedoch nur praktikabel, wenn genug Beispiele existieren, für die eine Klassenzugehörigkeit bekannt ist. In vielen Fällen, insbesondere wenn nur wenige Trainingsdaten zur Verfügung stehen, bietet sich eine Kreuzvalidierung zur Bewertung der Performanz eines Klassifikators an.

2.2.2.1 *K*-FACHE KREUZVALIDIERUNG

Um bei einer gegebenen Datenmenge X die Leistung eines Klassifikators abschätzen zu können, besteht ein einfacher Ansatz - wie oben beschrieben - darin, den Datenbestand in zwei disjunkte Untermengen zu teilen. Zum Einen in einen Trainingsdatensatz T , anhand dessen der Klassifikator aufgebaut bzw. trainiert wird, und zum Anderen in einen Testdatensatz V , der zur Ermittlung der Klassifikationsleistung verwendet wird. Dieses Vorgehen ist in der Praxis meist nicht einsetzbar, da der vorhandene Datenbestand oftmals keine ausreichende Größe aufweist²⁸. Eine alternative Vorgehensweise, die in der Praxis häufig zum Einsatz kommt und auch in dieser Arbeit angewendet wird, ist die *K-fache Kreuzvalidierung*. Hier wird der Datensatz X zufällig in K gleich große Teile X_i , $i=1,\dots,K$ zerlegt. Aus diesen Teilmengen lassen sich K Paare von Trainings- und Testdatensätzen $\{T_i, V_i\}_{i=1}^K$ gewinnen:

$$\begin{array}{ll} V_1 = X_1 & T_1 = X_2 \cup X_3 \cup \dots \cup X_K \\ V_2 = X_2 & T_2 = X_1 \cup X_3 \cup \dots \cup X_K \\ \vdots & \\ V_K = X_K & T_K = X_1 \cup X_2 \cup \dots \cup X_{K-1}. \end{array}$$

Für den Fall, dass die a priori-Wahrscheinlichkeiten der Zielklassen in allen Teilmengen X_i den a priori-Wahrscheinlichkeiten in X entsprechen, spricht man von *Stratifizierung*. Die Abschätzung der Klassifikationsleistung anhand von K Läufen führt in der Praxis üblicherweise zu genaueren Prognosen über die Klassifikationsleistung bisher ungesehener Daten. Ein Spezialfall der *K*-fachen Kreuzvalidierung ist das *Leave-One-Out-Verfahren*. Bei n vorhandenen Beispielen werden hier n Paare gebildet und die n an $n-1$ Instanzen aufgebauten Klassifikatoren werden an dem jeweils verbleibenden einzelnen Beispiel getestet. Eine tiefergehende Einführung in diese Thematik bieten u. a. [RN10] und [Alp08], aus dem die hier vorgestellte Notation übernommen wurde.

²⁸ Eine „ausreichende“ Größe ist von vielen Faktoren abhängig, insbesondere von der Dimensionalität der Eingabevektoren und der Anzahl der Zielklassen.

2.2.2.2 GÜTEMASSE

Um die Leistungsfähigkeit eines Klassifikators abzuschätzen und verschiedene Klassifikatoren für eine gegebene Problemstellung hinsichtlich ihrer Performanz vergleichbar zu machen, gibt es eine Reihe quantitativer Maßzahlen, die sich nach der Durchführung der Klassifikation an einer Testmenge oder aber einer K -fachen Kreuzvalidierung aus der Wahrheitsmatrix bzw. Konfusionsmatrix ableiten lassen. Für ein Klassifikationsproblem mit $Z > 2$ Klassen wird die Konfusionsmatrix auch Klassenkonfusionsmatrix genannt. Jedes Element c_{ij} repräsentiert die Anzahl der Instanzen, die zu einer Klasse C_j gehören, aber der Klasse C_i zugeordnet wurden. Im idealen Fall sind alle Nichtdiagonalelemente gleich 0, d.h. es wurden alle Testdaten korrekt klassifiziert. Im Folgenden werden die in dieser Arbeit verwendeten Metriken kurz erläutert.

SENSITIVITÄT

Die *Sensitivität*²⁹ einer Klasse C_i gibt den Anteil der korrekt C_i zugeordneten Testinstanzen an der Gesamtheit der Klasse C_i zugehörigen Instanzen an. Formal bedeutet dies:

$$S(C_i) = \frac{c_{ii}}{\sum_{j=1}^Z c_{ji}} \quad (2.9)$$

POSITIVER VORHERSAGEWERT

Der *positive Vorhersagewert*³⁰ einer Klasse C_i , auch *positiver prädiktiver Wert (PPV)* genannt, repräsentiert den Anteil der korrekt der Klasse C_i zugewiesenen Instanzen an der Gesamtheit der Klasse C_i zugewiesenen Instanzen. Dies bedeutet:

$$PPV(C_i) = \frac{c_{ii}}{\sum_{j=1}^Z c_{ij}} \quad (2.10)$$

²⁹ Auch Richtig-Positiv-Rate, Empfindlichkeit, Recall oder Trefferquote genannt.

³⁰ Synonym: Relevanz, Wirksamkeit, Genauigkeit, *Precision*.

KORREKTKLASSIFIKATIONSRATE

Die Korrektklassifikationsrate (KKR) beschreibt den Anteil aller korrekt klassifizierten Instanzen an der Gesamtheit aller Testinstanzen:

$$\text{KKR} = \frac{\sum_{i=1}^Z c_{ii}}{\sum_{i=1}^Z \sum_{j=1}^Z c_{ij}} \quad (2.11)$$

WURZEL DES MITTLEREN QUADRATISCHEN FEHLERS

Eine Maßzahl, die vor allem bei der Regression zur Bewertung der Qualität der Vorhersage zum Einsatz kommt, ist die Wurzel des mittleren quadratischen Fehlers, auch als *Root Mean Square Error (RMSE)* bezeichnet. Da die Hauptaufgabe dieser Arbeit darin besteht, zu untersuchen, wie gut der CART-Algorithmus die gegnerischen Aktionen vorherzusagen vermag - und dies geschieht in Form von Wahrscheinlichkeiten - bietet sich der RMSE auch hier zur Analyse der Performanz an. Für eine Menge von n Testdatensätzen, wobei p_i der Vorhersagewert und a_i der tatsächlich beobachtete Wert der i -ten Instanz ist, errechnet sich der RMSE wie folgt:

$$\text{RMSE} = \sqrt{\frac{\sum_{i=1}^n (p_i - a_i)^2}{n}} \quad (2.12)$$

Da in vorliegender Aufgabenstellung drei Zielklassen mit Wahrscheinlichkeiten belegt werden, ist vor allem von Interesse, welche Wahrscheinlichkeiten den beiden falschen Klassen für eine Instanz i zugewiesen werden. Es bezeichne W_{if} die Summe dieser Wahrscheinlichkeiten, dann lässt sich der RMSE, so wie er im weiteren Verlauf dieser Arbeit berechnet wird, folgendermaßen notieren:

$$\text{RMSE} = \sqrt{\frac{\sum_{i=1}^n W_{if}^2}{n}} \quad (2.13)$$

2.3 FAZIT

In diesem Kapitel wurden die zum weiteren Verständnis der Arbeit benötigten Grundlagen gelegt. Trotz relativ einfacher Regeln ist Texas Hold'em strategisch komplex und bietet für die Forschung noch viele Herausforderungen. Unvollständige Information, Zufallsereignisse und die Eigenschaft der nur teilweisen Beobachtbarkeit der Spielzustände sind Faktoren, die die Entwicklung eines spielstarken Pokersystems zu einer großen Herausforderung machen. Zwar wurden durch spieltheoretische Ansätze sehr spielstarke Systeme erzeugt, doch ist es ihnen nicht möglich, offensichtlich erkennbare Schwächen des Gegners auszunutzen. Ansätze, die auf einer Spielbaumsuche beruhen, und hier insbesondere die Varianten Minimax und Maximax, sind dazu zwar in der Lage, jedoch hängt die Spielstärke dieser Systeme direkt von der Güte des verwendeten Gegnermodells ab.

Zur Erstellung eines solchen Modells scheint sich der CART-Algorithmus, ein Vertreter der Entscheidungsbäume, gut zu eignen. CART kann als relativ robust gegenüber Ausreißern angesehen werden und eine aufwendige Vorverarbeitung und Selektion der Eingaben entfällt. Insbesondere die Interpretierbarkeit der von CART erzeugten Bäume und das Vorgehen beim Klassifizieren der Instanzen, das ähnlich einer menschlichen Entscheidungsfindung stattfindet, bieten interessante Möglichkeiten für zukünftige Arbeiten.

Um die Leistung des CART-Algorithmus im weiteren Verlauf dieser Arbeit zu bewerten, wurden in diesem Kapitel die KKR und der RMSE eingeführt. Für einen Blick auf die Qualität der Klassifikation der einzelnen Klassen bieten sich darüber hinaus die Sensitivität und der positive Vorhersagewert an.

3 DIE ERSTELLUNG DES GEGNERMODELLS

„Spielen ist eine Tätigkeit, die man gar nicht ernst genug nehmen kann.“

Jacques-Yves Cousteau

Das Problem der Modellierung der Gegenspieler beim Texas Hold'em ist äußerst komplex. Je mehr Spiele eines Gegners beobachtet werden, desto besser ist ein Mensch in der Lage, diesen einzuschätzen. Sehr gute menschliche Spieler vermögen ihren Gegner bereits nach wenigen Spielen relativ genau zu beurteilen. Dies geschieht analog zur Klassifikation, wobei jeder menschliche Spieler seine ihm eigenen Regeln und Vorgehensweisen besitzt, wie er seine Gegenspieler in Klassen wie z.B. „*aggressiver Spieler*“, „*Anfänger*“ oder „*Calling-Station*“³¹ einordnet. Auch bei der Konstruktion eines adaptiven Pokersystems kommt der Modellierung des Gegenspielers, d.h. dem Gegnermodell, eine entscheidende Bedeutung zu. Zwar ist die Aufgabenstellung dieser Arbeit die Erstellung und Bewertung des Teilmodells zur Vorhersage gegnerischer Aktionen, dennoch soll hier das Gegnermodell auch im Ganzen betrachtet werden.

Teile dieses Kapitels können als eine Erweiterung des vorangegangenen Kapitels verstanden werden, da sie Grundlagen aus dem Bereich der Gegnermodellierung beschreiben. Dennoch sind diese Grundlagen hier zu finden, da dieses Kapitel vollständig dem Aufbau, Einsatz und der Erstellung des Gegnermodells gewidmet ist.

Zunächst werden Aufbau und Einsatz des Gegnermodells in der Spielbaumsuche beschrieben. Im Anschluss daran wird die Struktur der hier verwendeten Merkmalsvektoren vorgestellt. Darauf folgt ein kurzer Vergleich der Klassifikationsleistung von CART zu einem

³¹ Als „Calling-Station“ bezeichnet man einen Spieler, der ungeachtet der eigenen Hand fast immer mitgeht. Dies ist offensichtlich ein sehr schwacher Spieler.

KNN, bevor die Erzeugung der Datenbasis dieser Arbeit näher erläutert wird und die verwendeten Werkzeuge vorgestellt werden.

3.1 AUFBAU UND EINSATZ DES GEGNERMODELLS

Die Aufgabe des Gegnermodells in der Spielbaumsuche ist, an den gegnerischen Entscheidungsknoten Wahrscheinlichkeiten für die möglichen gegnerischen Aktionen bereitzustellen. Darüber hinaus ist es Aufgabe des Modells, an Spielendknoten, die einen Showdown darstellen, Wahrscheinlichkeiten für einen Sieg zu liefern.

Für jede Setzrunde und zur Schätzung der Erwartungswerte an den Spielendknoten werden dabei eigene Teilmodelle, d.h. von CART aufgebaute Klassifikationsbäume, verwendet. Diesen Sachverhalt verdeutlicht Abbildung 3.1.

Nehmen wir an, wir befinden uns in der Spielbaumsuche nach einer gegnerischen Erhöhung an einem eigenen Entscheidungsknoten, der als Kreis dargestellt ist. Wir interessieren uns für die Wahrscheinlichkeiten der gegnerischen Aktionen, für den Fall, dass wir eine Erhöhung tätigen. Von diesem Knoten führen drei Zweige - für die drei zur Verfügung stehenden Aktionen *Fold* (F)³², *Call* (C) und *Raise* (R) - zu seinen drei Kindknoten. Teilbäume, die durch einen Call unsererseits oder des Gegners betreten werden, sind als Dreieck dargestellt, aber hier nicht von Bedeutung. Der gegnerische Entscheidungsknoten ist durch ein Quadrat symbolisiert. Die Sechsecke repräsentieren Spielendknoten, die durch das Aussteigen aus dem Spiel erreicht werden.

Vom gegnerischen Entscheidungsknoten wird nun das Gegnermodell unter Angabe des vollständigen Spielkontextes aufgerufen. Dieser umfasst die vollständige Setz-Sequenz, die bis zu diesem Knoten beobachtet wurde und darüber hinaus die Gemeinschaftskarten. Aus diesem Spielkontext generiert das Gegnermodell einen Merkmalsvektor, dessen Aufbau in 3.1.1 beschrieben ist. Dieser Vektor wird an das Teilmodell, d.h. den entsprechenden Klassifikationsbaum für die jeweilige Setzrunde, weitergeleitet und klassifiziert. Wie in 2.2.1

³² Gemäß der in 2.1.4 eingeführten Notation werden eigene Aktionen in Großbuchstaben angegeben, während gegnerische Aktionen in Kleinbuchstaben notiert sind.

beschrieben, liefert der entsprechende Baum dann die Wahrscheinlichkeiten für die zur Verfügung stehenden gegnerischen Aktionen.

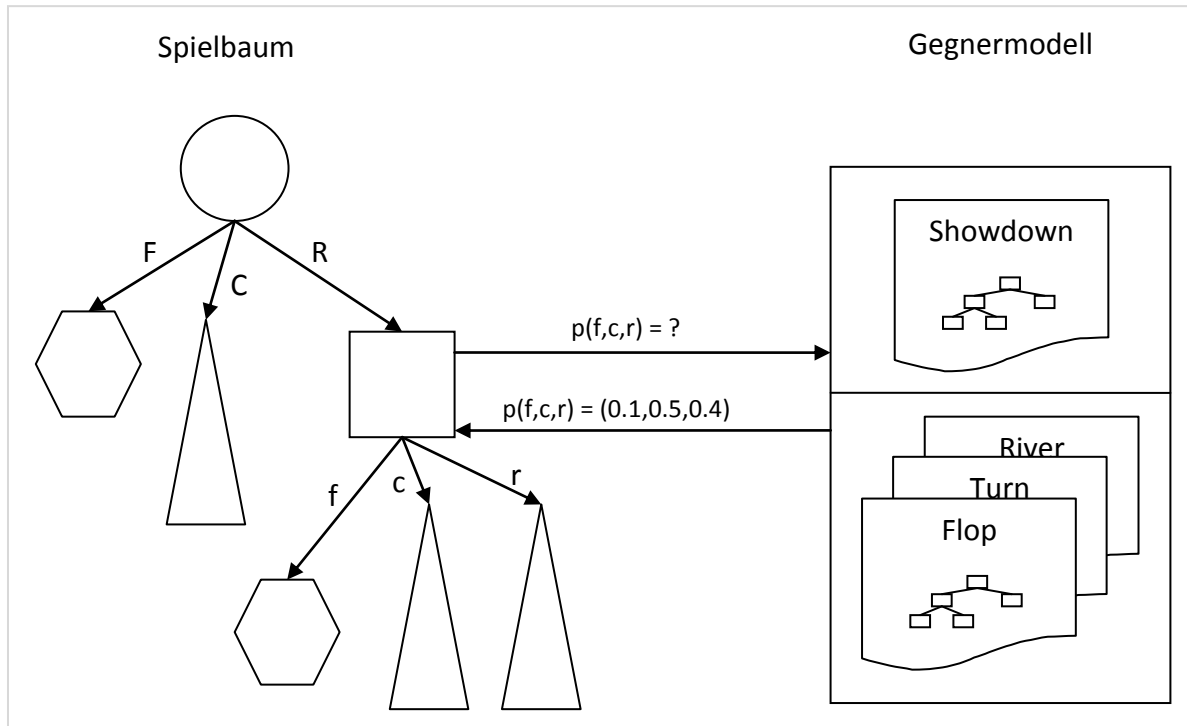


Abbildung 3.1: Einsatz des Gegnermodells in der Spielbaumsuche. Die Abbildung zeigt die Anfrage an das Modell an einem gegnerischen Entscheidungsknoten. Der Spielkontext am gegnerischen Entscheidungsknoten wird vom Gegnermodell in einen Merkmalsvektor umgewandelt, der dann dem entsprechenden Klassifikationsbaum als Eingabe übergeben wird.

Nach jedem beendeten Spiel werden aus der Setz-Sequenz und den Gemeinschaftskarten Merkmalsvektoren gemäß Abschnitt 3.2 für alle Teilmodelle erzeugt, die dann den bestehenden Trainingsmengen für die einzelnen Setzrunden hinzugefügt werden. Im Anschluss daran werden für alle beobachteten Setzrunden³³ dieses Spiels die Klassifikationsbäume anhand der neuen Trainingsmengen erneut aufgebaut. Das Vorgehen für das Teilmodell zur Approximation der Erwartungswerte an den Spielendknoten erfolgt analog, jedoch finden hier noch die gegnerischen Karten Eingang in die Erzeugung der Merkmalsvektoren.

³³ Für den beispielhaften Fall, dass ein Spiel bereits auf dem Flop endet, existieren natürlich keine Beobachtungen für den Turn und River.

Erwähnenswert ist die Tatsache, dass der hier beschriebene Fall nur eine Möglichkeit der Gegnermodellierung darstellt. Billings et al. unterscheiden in [BDSS02] zwei verschiedene Ansätze. Das Vorgehen, ein Gegnermodell anhand der Beobachtungen zu erstellen, die nur für diesen Gegner gemacht wurden, wird *spezifische Gegnermodellierung (specific opponent modeling)* genannt. Wird dem Gegner dagegen eine bestimmte rationale Spielweise unterstellt, so könnte ein allgemeines Gegnermodell zum Einsatz kommen, das anhand der Beobachtungen für viele verschiedene Spieler erstellt wurde. Dies wird auch als *generische Gegnermodellierung (generic opponent modeling)* bezeichnet.

Darüber hinaus führte van der Kleij in [van10] den Begriff der *gruppenspezifischen Gegnermodellierung (group-specific opponent modeling)* ein. Hier werden verschiedene Gegnermodelle für bestimmte Spieler-Typen generiert und eingesetzt.

3.1.1 Aufbau der Merkmalsvektoren

Die in der vorliegenden Arbeit verwendeten Merkmalsvektoren umfassen das Klassenlabel und 22 weitere Attribute und beruhen auf den in [van10] und [Dav99] vorgeschlagenen Merkmalen. Tabelle 2 gibt einen Überblick über die einzelnen Merkmale und erklärt diese. Anzumerken ist, dass nicht alle Merkmale für jede Setzrunde Verwendung finden. Beispielsweise stehen auf dem Turn noch keine Informationen über den River zur Verfügung. Mit „*Spieler*“ wird in folgender Tabelle der zu modellierende Spieler bezeichnet. Sein Gegner trägt die Bezeichnung „*Gegner*“.

Tabelle 2: Übersicht und Erklärung der verwendeten Merkmale, die zur Beschreibung der Spielkontexte verwendet werden.

Bezeichnung	Typ	Erklärung
potSize	numerisch	Größe des Pots.
hasToCallBet	boolesch	True, falls der Gegner in dieser Runde zuvor einen Einsatz getätigt hat.
numberOfActionsThisRound	numerisch	Anzahl der bereits getätigten Aktionen in der gegenwärtigen Setzrunde.

Bezeichnung	Typ	Erklärung
lastActionByPlayerWasBetOrRaise	boolesch	True, falls die letzte Aktion des Spielers ein Bet oder Raise war. Dies kann auch in der vorangehenden Runde passiert sein.
lastActionByPlayer	nominal	Die genaue Bezeichnung der letzten Aktion des Spielers.
numberOfBetsAndRaisesPreflop	numerisch	Anzahl aller Bets und Raises vor dem Flop.
numberOfBetsAndRaisesFlop	numerisch	Anzahl aller Bets und Raises auf dem Flop.
numberOfBetsAndRaisesTurn	numerisch	Anzahl aller Bets und Raises auf dem Turn.
numberOfBetsAndRaisesRiver	numerisch	Anzahl aller Bets und Raises auf dem River.
wasAggressorPreflop	boolesch	True, wenn der Spieler die letzte Erhöhung oder den letzten Einsatz vor dem Flop getätigt hat.
wasAggressorFlop	boolesch	True, wenn der Spieler der Aggressor auf dem Flop war/ist.
wasAggressorTurn	boolesch	True, wenn der Spieler der Aggressor auf dem Turn war/ist.
wasAggressorRiver	boolesch	True, wenn der Spieler der Aggressor auf dem River ist.
maxNumberOfSuitedCardsFlop	numerisch	Die größte Anzahl der Karten gleicher Farbe auf dem Flop.
maxNumberOfSuitedCardsTurn	numerisch	Die größte Anzahl der Karten gleicher Farbe auf dem Turn.
maxNumberOfSuitedCardsRiver	numerisch	Die größte Anzahl der Karten gleicher Farbe auf dem River.
averageCardRankForCurrentRound	numerisch	Durchschnittlicher Rang der Gemeinschaftskarten. Eine Zwei hat den Rang 0. Die Karten sind aufsteigend bis zum Ass mit Rang 12 geordnet.
maxNumberOfSameRankForCurrentRound	numerisch	Die größte Anzahl der Karten gleichen Ranges.

Bezeichnung	Typ	Erklärung
numberOfDifferentHighCardsForCurrentRound	numerisch	Die Anzahl der Gemeinschaftskarten, die höher als eine Zehn sind.
highestNumericCardRankFlop	numerisch	Der Rang der höchsten Karte auf dem Flop.
highestNumericCardRankTurn	numerisch	Der Rang der höchsten Karte auf dem Turn.
highestNumericCardRankRiver	numerisch	Der Rang der höchsten Karte auf dem River.

Abschnitt 3.2 beschreibt das genaue Vorgehen, wie aus einem abgeschlossenen Spiel Entscheidungspunkte bzw. Beobachtungen für die einzelnen Setzrunden gewonnen werden können. Aus diesen Beobachtungen, die jeweils den vollständigen Spielkontext enthalten, lassen sich die Trainingsdaten gewinnen, aus denen die Entscheidungsbäume für die jeweiligen Setzrunden generiert werden.

3.1.2 Weitergehende Konzepte für den Einsatz in der Praxis

Um eine Bewertung des hier vorgestellten Gegnermodells zu ermöglichen, werden viele Spiele betrachtet und notwendigerweise Vereinfachungen und Abstraktionen vorgenommen. Zur Erstellung eines spielstarken Systems sind weitere Verfeinerungen des Modells notwendig. Im Folgenden wird die Komplexität der sich stellenden Probleme anhand der geforderten Adaptionfähigkeit eines Gegnermodells verdeutlicht. Eine eingehende Untersuchung weiterer Probleme wäre sicherlich für weitere Arbeiten interessant.

ADAPTIONSFÄHIGKEIT

Ganz gleich in welcher Variante der Baumsuche das hier vorgeschlagene Gegnermodell zum Einsatz kommt, ist es wichtig, dass sich das Modell dem sich ändernden Spielverhalten des zu modellierenden Spielers anpasst. Zur Bewertung der erreichbaren Performanz des Modells, werden in dieser Arbeit pro Spieler zum Teil tausende Spiele betrachtet. Um aber

Strategiewechsel der Gegner zu modellieren, ist es notwendig, die Anzahl der Trainingsdaten, anhand derer die Entscheidungsbäume aufgebaut werden, zu begrenzen. Eine wichtige Frage in diesem Zusammenhang ist, welche Größe die Trainingsmenge haben soll, um verlässliche Schätzungen abgeben zu können. Wird die Menge zu klein gewählt, ist die Klassifikationsleistung des CART-Algorithmus schlecht. Im Falle einer zu großen Trainingsmenge gehen Strategieänderungen sozusagen in der Basisstrategie unter. Aus diesem Grund wird im weiteren Verlauf dieser Arbeit unter Anderem untersucht, wie schnell der CART-Algorithmus in der Lage ist, verlässliche Ergebnisse zu liefern. Die gewonnenen Erkenntnisse könnten dazu verwendet werden, das hier entwickelte Modell zu verfeinern. Ein vielversprechender Ansatz scheint auch die Kombination mehrerer Entscheidungsbäume zu sein, die anhand von Trainingsdatensätzen aufgebaut werden, die sich hinsichtlich ihrer Größe unterscheiden. Zum Beispiel könnte ein Baum anhand aller beobachteten Spiele erstellt werden um das grundlegende Spielverhalten des Gegners abzubilden³⁴. Ein weiterer Baum, der nur auf der Grundlage relativ weniger Beobachtungen erstellt wurde, könnte dazu verwendet werden, die augenblicklich gewählte Strategie des Gegners zu modellieren. Fragen die sich daraus ergeben sind z.B. die Gewichtung der einzelnen Bäume und insbesondere auch deren Anzahl, da sich diese Vorgehensweise prinzipiell beliebig verfeinern lässt. Begrenzt wird dieser Ansatz in natürlicher Weise durch die zur Verfügung stehende Rechenleistung, da die Spielbaumsuche in jeder gegebenen Situation nicht mehr als wenige Sekunden in Anspruch nehmen sollte.

3.2 ERSTELLUNG DER DATENBASIS

Zur Unterstützung der Entscheidungsfindung bzw. zur Gegnermodellierung für den Menschen existieren schon seit längerer Zeit Tools, die es erlauben, Spiele in Internet-Poker-Plattformen mit zu protokollieren. Das Interessante daran ist, dass diese Tools in der Lage sind, beliebig viele Tische zu beobachten. So ist es möglich, dass ein menschlicher Spieler

³⁴ Falls noch keine Informationen über den Gegner zur Verfügung stehen, könnte auch ein Baum, der anhand der Daten anderer Spieler aufgebaut wird, zum Einsatz kommen. Dies entspräche dann einer Verschmelzung von generischer und spezifischer Gegnermodellierung.

gegen einen anderen Spieler antritt und bereits Informationen über diesen Spieler besitzt, da zuvor potenziell viele tausend Spiele dieses Spielers beobachtet wurden.

Da diese historischen Spieldaten bzw. die Informationen über bestimmte Spieler sehr „wertvoll“ sind – es geht um echtes Geld – existieren Firmen, die gewerbsmäßig alle Spiele auf den bekannten Online-Poker-Plattformen mitprotokollieren und diese Daten dann zum Kauf anbieten.

Für diese Arbeit wurde ein Datensatz von 400.000 Händen erworben, die alle auf derselben Online-Poker-Plattform gespielt wurden. Alle Spiele wurden mit den gleichen Einsätzen von 1\$/2\$ gespielt. Dies bedeutet, dass ein Small Bet die Höhe von 1\$ beträgt, ein Big Bet hingegen 2\$. Dieses Vorgehen wurde gewählt, da hier unterstellt wird, dass Spiele, die um richtiges Geld gespielt werden, für diese Arbeit interessanter sind, als Spiele, die ohne Einsätze stattfinden und nur zum Spaß gespielt werden. Beispielsweise lässt sich in Spielen ohne Einsatz ein viel aggressiveres Verhalten der Spieler beobachten und Showdowns finden häufiger statt. Problematisch an diesem Ansatz ist, dass die Spiele in einem Zeitraum eines Jahres gesammelt wurden. Das bedeutet vor allem, dass die Spiele der einzelnen Spieler gegen viele verschiedene Gegner gespielt wurden. Da sich Spieler aber mit ihrer Strategie an ihre Gegner anpassen, d.h. ihre Spielweise ändern, wird sich dieser Sachverhalt möglicherweise negativ auf die Klassifikationsleistung auswirken. Trotzdem wurde hier dieser Ansatz gewählt, da es äußerst schwierig ist, gute menschliche Spieler für eine Serie von mehreren tausend Spielen gegen einen Computer oder einen anderen menschlichen Gegner zu gewinnen. Darüber hinaus ändern gute Spieler im Verlauf einer Serie von Spielen gegen denselben Gegner ebenso ihre Strategie, worauf der Gegner dann seine Strategie ändert und so fort.

Die Daten liegen in 407 Textdateien vor, die jeweils ca. 1.000 Spiele in Textform beinhalten. Durch ein Java-Programm wurden zunächst die 20 Spieler mit den meisten Spielen ermittelt, aus denen dann zufällig 10 Spieler ausgewählt wurden, die die Datengrundlage dieser Arbeit bilden.

Da hier der Ansatz verfolgt wird, getrennte Entscheidungsbäume für den Flop, Turn und River zu entwickeln, wurden die Spiele zunächst in diese Runden aufgespalten. Beispielsweise lässt sich ein Spiel mit der Satz-Sequenz *cK/BrRrC/Bc/BrF*, gemäß der

Notation, die in 2.1.4 eingeführt wurde, aufspalten in $cK/BrRrC$ für den Flop, $cK/BrRrC/Bc$ für den Turn und $cK/BrRrC/Bc/BrF$ für den River.

Aus den so aufgespaltenen Spielen wurden in einem zweiten Schritt dann alle Beobachtungen über die Aktionen des zu modellierenden Spielers extrahiert. Für den Flop lassen sich zwei Beobachtungen bzw. Entscheidungspunkte gewinnen. Diese Entscheidungspunkte sind die Situationen, in denen der zu modellierende Spieler eine Entscheidung treffen muss hinsichtlich seiner nächsten Aktion. Die Sequenz $cK/BrRrC$ wird weiter zerlegt in cK/Br und $cK/BrRr$. Die jeweils letzten Buchstaben beider Entscheidungspunkte sind die vorauszusagenden Aktionen, hier beispielsweise ein *Raise*. Aus der restlichen Setz-Sequenz lässt sich dann der Spielkontext, z.B. die Größe des Pots oder aber das Aggressionsverhalten beider Spieler, gewinnen.

Zusammen mit den in den jeweiligen Spielrunden bekannten Board Cards liefern dann diese Entscheidungspunkte die Grundlage zur Gewinnung der Inputs bzw. der Merkmale, die dem CART-Algorithmus anschließend als Eingabe für die Klassifizierung dienen. Die Namen der Spieler wurden anonymisiert, die Zuordnung zu den realen Namen findet sich auf der dieser Arbeit beiliegenden CD-ROM.

3.2.1 Struktur der Daten

Einen Überblick über die Anzahl der Spiele, aus denen die hier verwendete Datenbasis gewonnen wurde, gibt Tabelle 3.

Tabelle 3: Anzahl der Spiele und Entscheidungspunkte für die hier betrachteten Spieler.

Name	Anzahl der Spiele	Flop Entscheidungs- punkte	Turn Entscheidungs- punkte	River Entscheidungs- punkte
Spieler1	1995	643	638	432
Spieler2	1740	687	655	441
Spieler3	4897	1625	1871	1305
Spieler4	2379	891	1039	715
Spieler5	2883	1260	1391	987

Name	Anzahl der Spiele	Flop Entscheidungs- punkte	Turn Entscheidungs- punkte	River Entscheidungs- punkte
Spieler6	3004	1627	1728	1102
Spieler7	9009	4564	4300	3155
Spieler8	2852	1612	1541	1043
Spieler9	25375	10508	10609	7450
Spieler10	2598	1132	1154	781
Summe	56732	24549	24926	17411

Für sechs der zehn Spieler stehen mehr Beobachtungen für den Turn zur Verfügung als für den Flop. Dies erklärt sich daher, dass alle Situationen, in denen der zu modellierende Spieler als Erster am Flop zu handeln hat, herausgefiltert wurden. Dies geschah, da diese Situationen in der hier betrachteten Modellierung im zugehörigen Spielbaum nie auftauchen bzw. vorherzusagen sind.

Abbildung 3.2 verdeutlicht die Verteilung der relativen Häufigkeiten der möglichen Aktionen am Turn für die zehn betrachteten Spieler. Für alle Spieler gilt die Tatsache, dass die relativen Häufigkeiten für einen Fold sehr viel geringer sind als für einen Check oder einen Raise. Die Wahrscheinlichkeiten für einen Fold am Turn liegen zwischen 0,0752 für Spieler1 und 0,152 für Spieler10. Diese ungleichmäßige Verteilung wird sich unmittelbar auf die Klassifikationsleistung hinsichtlich dieser Minderheitsklasse auswirken und in Kapitel 4 untersucht. Die Klassenverteilungen für den Flop und River finden sich in auf der beiliegenden CD-ROM. Betrachtet man die relativen Häufigkeiten der einzelnen Aktionen für die verschiedenen Setzrunden für die einzelnen Spieler, dann fällt auf, dass die einzelnen Spieler in den verschiedenen Setzrunden offensichtlich unterschiedlich spielen. Dies kann als Indiz dafür gewertet werden, dass es besser ist einzelne Teilmodelle für die verschiedenen Setzrunden zu entwickeln, als ein Gesamtmodell für alle Runden gemeinsam.

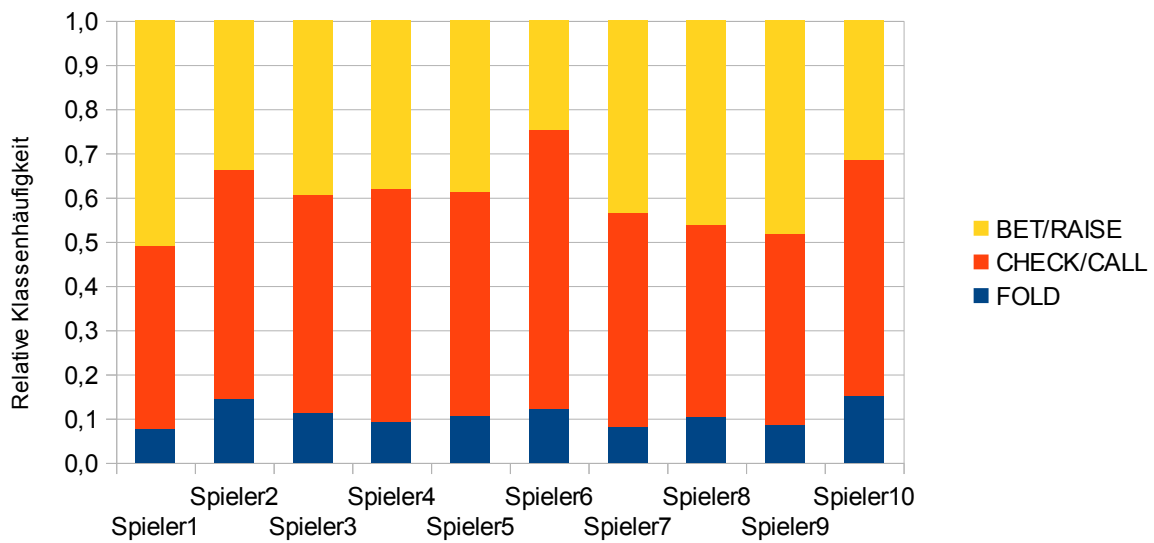


Abbildung 3.2: Die Verteilung der relativen Häufigkeiten der möglichen Aktionen für den Turn. Auffällig ist, dass die relativen Häufigkeiten für einen Fold, im Vergleich zu denen der anderen Aktionen, sehr gering sind.

3.3 EXPLORATIVE EVALUIERUNG DER PERFORMANZ IM VERGLEICH ZU EINEM KNN

Um einen ersten Überblick über die Vorhersagequalität des CART-Algorithmus im Kontext der hier bearbeiteten Aufgabenstellung zu gewinnen, wird im Folgenden ein Blick auf die Klassifikationsleistung für einen in sich abgeschlossenen größeren Datensatz geworfen. In sich abgeschlossen bedeutet hier, dass als Datengrundlage die Beobachtungen einer einzelnen Setzrunde für einen einzigen Spieler betrachtet werden. Die zahlenmäßig größte Datengrundlage, die auch im weiteren Verlauf dieses Kapitels exemplarisch betrachtet wird, bietet der Turn für Spieler9 mit 10609 Beobachtungen. Um die Klassifikationsleistung von CART besser beurteilen zu können, wird sie mit der eines KNN verglichen.

Der Vergleich zu einem KNN bietet sich hier besonders an, da Davidson in [Dav99] ein KNN zur Vorhersage der gegnerischen Aktionen für das Limit Hold'em mit mehr als zwei Spielern verwendet und anhand der KKR analysiert. Das Spiel mit mehr als zwei Spielern unterscheidet sich zwar grundlegend von der Heads-Up Variante, insbesondere sind Folds sehr viel häufiger zu beobachten, doch wurde in dieser Arbeit die Möglichkeit verdeutlicht, künstliche neuronale Netze zur Gegnermodellierung einzusetzen.

Auf eine Darstellung der Arbeitsweise künstlicher neuronaler Netze sei an dieser Stelle verzichtet. Eine Einführung findet sich beispielsweise in [Alp08] und [RN10].

3.3.1 Aufbau des KNN

Bei dem hier durchgeführten Performanzvergleich kommt ein mehrschichtiges feedforward-Netz zum Einsatz, das anhand des Backpropagation-Algorithmus trainiert wird. Das Netz besitzt eine verdeckte Schicht mit 14 Neuronen und ist vollverknüpft. Als Aktivierungsfunktion wird eine Sigmoidfunktion gewählt. Die drei vorherzusagenden Klassen werden durch drei Ausgabeneuronen repräsentiert, deren Ausgabe jeweils im Intervall $[0,1]$ liegt.

Die nominalen Eingaben, mit denen ein KNN im Gegensatz zu CART nicht umgehen kann, werden in numerische Werte umgewandelt. Dabei werden die zweiwertigen nominalen Eingaben jeweils auf 0 bzw. 1 abgebildet und *lastActionByPlayer* (siehe Tabelle 2), das vier Merkmalsausprägungen annehmen kann, wird auf vier binäre Eingabeneuronen abgebildet, sodass die Eingabeschicht letztendlich 25 Neuronen umfasst. Alle Eingaben werden zudem normalisiert. Die Lernrate des KNN wird auf 0,3 festgesetzt, für das Momentum wird ein Wert von 0,2 gewählt. Das Netz wird 500 Epochen bzw. Zyklen trainiert.

3.3.2 KKR und RMSE

Abbildung 3.3 zeigt die grafische Repräsentation des Ergebnisses einer stratifizierten 10-fachen Kreuzvalidierung für den CART-Algorithmus und das KNN hinsichtlich der KKR und des RMSE. Die Ergebnisse der 10 einzelnen Durchläufe sind über die x-Achse aufgetragen, wobei die Werte des CART-Algorithmus durch Balken und die Werte des KNN durch Linien dargestellt sind.

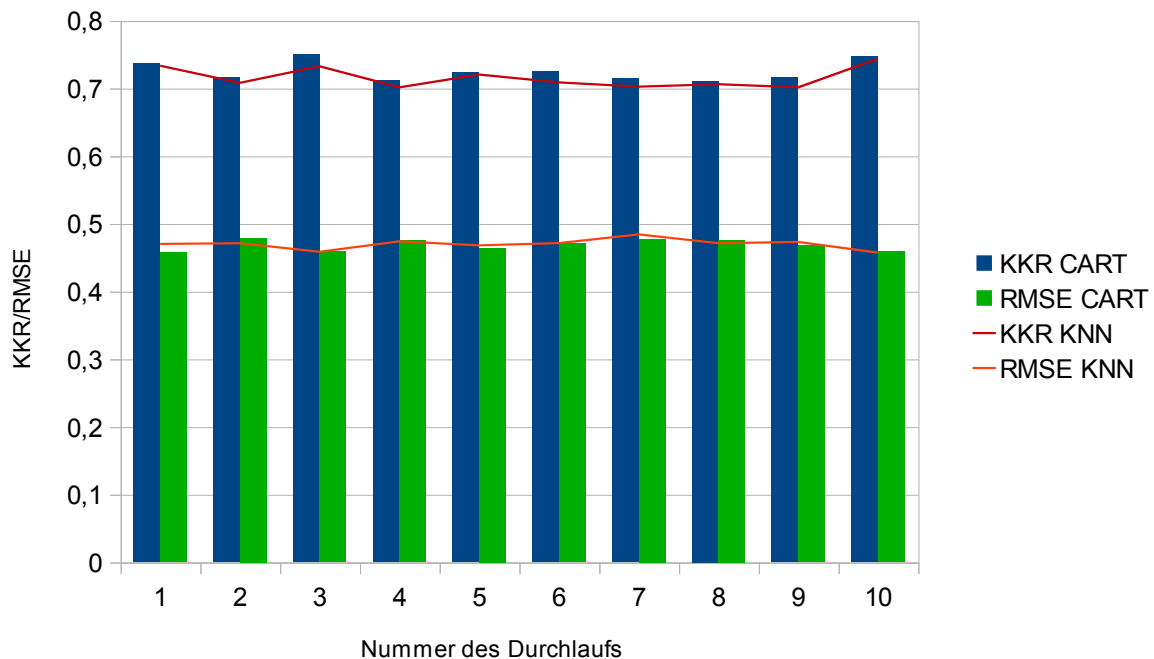


Abbildung 3.3: Vergleich der Klassifikationsleistung von CART zu der eines KNN anhand der Daten von Spieler9 für den Turn. Die Leistung beider Verfahren für die gegebenen Daten ist vergleichbar.

Die KKR und der RMSE für beide Klassifikatoren liegen sehr nahe beieinander. CART erreicht eine mittlere KKR von 0,725. Das KNN erreicht im Mittel eine KKR von 0,716. In keinem der 10 Durchläufe erreicht das KNN eine bessere KKR als CART, allerdings konnte das KNN in fünf Durchläufen einen geringeren RMSE erzielen als CART. Anzumerken ist, dass für einen gepaarten Zweistichproben-t-Test auf einem Konfidenzniveau von 0,95 davon ausgegangen werden kann, dass sich die Korrektclassifikationsraten und RMSE des CART-Algorithmus und des KNN für diesen Fall nicht signifikant unterscheiden³⁵.

Die Probleme, die sich im Hinblick auf Signifikanztests bei der Bewertung von Klassifikatoren ergeben, sind ausführlich in [JM11] dargestellt.

³⁵ Um dies zu prüfen, wurde eine 35-fache Kreuzvalidierung durchgeführt, da hier keine Normalverteilung der gepaarten Differenzen (betrachtet wurden hier die Ergebnisse der einzelnen Durchläufe) angenommen werden kann, die für Stichproben von weniger als 30 Elementen Voraussetzung zur Durchführung des t-Tests ist. Dieses Vorgehen ist in [Alp08] beschrieben. Details zur Durchführung des t-Tests finden sich auf beiliegender CD-ROM.

3.3.3 Stabilität

Eine wichtige Beobachtung ist, dass die Performanz des KNN, wie sie in Abbildung 3.3 dargestellt ist, den optimalen Fall hinsichtlich der Stabilität darstellt. Vereinzelt liegt die KKR des KNN für einen einzelnen Durchlauf der Kreuzvalidierung mit bis zu 0,15 deutlich unter der KKR des CART-Algorithmus. Dies lässt sich dadurch erklären, dass das KNN in der Trainingsphase einem *lokalen Minimum* nicht mehr entkommen kann.

Im Gegensatz dazu wurde im Verlauf dieser Arbeit ein derartiges Verhalten des CART-Algorithmus nicht beobachtet. Allgemein stellt dieses Stabilitätskriterium bei der Auswahl eines Klassifikators zur Gegnermodellierung ein wichtiges Kriterium dar, das insbesondere von CART erfüllt wird.

3.3.4 Zeitliche Performanz

Eine weitere wichtige Dimension hinsichtlich der Performanz ist die Zeit. Während CART prinzipiell nicht parametrisiert ist, und die Zeit für den Aufbau und das Pruning des Klassifikationsbaums von den Trainingsdaten abhängt, wird die Zeit zum Trainieren des KNN hauptsächlich von der Anzahl der Trainingszyklen bestimmt. Die Anzahl von 500 Trainingszyklen wurde sehr großzügig bemessen. Tests mit mehr als 500 Zyklen führten nicht zu besseren Ergebnissen als in Abbildung 3.3 dargestellt. Testläufe mit weniger Zyklen führten überwiegend zu vergleichbaren Ergebnissen, jedoch nicht verlässlich, sodass zur groben Abschätzung der KKR und des RMSE die Zahl von 500 Zyklen gewählt wurde. Die hier und im weiteren Verlauf der Arbeit verwendete Hardware war ein Intel Core2 Duo Prozessor mit 2,4 GHz Taktfrequenz und 4 GB Arbeitsspeicher. Für die Durchführung wurde nur ein Prozessorkern verwendet. Die hier ermittelten Zeiten sind eher als grobes Verhältnis der zeitlichen Performanz der beiden Verfahren zu verstehen. Die absoluten Zeiten werden nur der Vollständigkeit halber genannt und skalieren in hohem Maße mit der zur Verfügung stehenden Hardware.

Für den Aufbau und das Pruning des Klassifikationsbaumes anhand der 9548 Trainingsbeispiele benötigte CART im Mittel 4,28 Sekunden, der Median lag bei 4,11 Sekunden. Im Gegensatz dazu liegt der Mittelwert der Trainingszeit des KNN bei 60,72 Sekunden, der Median bei 60,61 Sekunden. Daraus ergibt sich ein Verhältnis der mittleren

Trainingsdauer des KNN im Vergleich zu CART von 14,18. Das Verhältnis für den Median beträgt 14,73.

Zur Klassifikation der 1061 verbleibenden Testinstanzen benötigte der CART-Algorithmus im Mittel 12,6 Millisekunden, der Median lag bei 4ms³⁶. Das KNN war erneut deutlich langsamer mit durchschnittlich 69,4ms und einem Median von 42,5ms. Für den Mittelwert ergibt sich daraus ein Faktor von 5,51 für den Mittelwert und 10,63 für den Median.

3.3.5 Fazit

Während die KKR und der RMSE für CART und ein relativ generisches KNN oftmals sehr nahe beieinander liegen, kann die Stabilität der Performanz für das KNN nicht garantiert werden. Zwar sind Ansätze denkbar, die diese Problematik zumindest abmildern, doch wären weitere Untersuchungen hierzu notwendig. Beispielsweise bietet sich hier der Einsatz eines gegenüber lokalen Minima robusteren Lernalgorithmus, z.B. *Resilient Propagation*, der in [RB92] beschrieben wird, an.

Eine Betrachtung der zeitlichen Performanz lässt Zweifel daran aufkommen, ob ein Einsatz eines KNN - zumindest in der hier vorgeschlagenen Modellierung – für die adaptive Spielbaumsuche praktikabel ist³⁷. Hierzu wären weitere Tests mit schnelleren Implementierungen notwendig.

3.4 DIE WERKZEUGE

Im Folgenden werden die zur Erstellung dieser Arbeit verwendeten Werkzeuge und Programme kurz beschrieben.

³⁶ Einer der 10 Durchläufe von CART war mit 87 ms ein Ausreißer. Die restlichen Durchläufe benötigten alle 4 bzw. 5ms.

³⁷ Bei einer Monte-Carlo Baumsuche kann nach einer vorgegebenen Zeit abgebrochen werden. Das Ergebnis ist aufgrund einer geringeren Anzahl an Simulationsläufen dann in der Regel weniger genau. Bei der adaptiven Spielbaumsuche muss der Baum komplett aufgebaut werden (siehe 2.1.5.3).

3.4.1 WEKA

WEKA steht für *Waikato Environment for Knowledge Analysis* und ist ein Softwaretool, das eine Vielzahl an Algorithmen aus den Bereichen des maschinellen Lernens und Data-Minings beinhaltet. Entwickelt wurde es an der *Universität von Waikato in Hamilton, Neuseeland*. WEKA ist vollständig in Java implementiert und kostenlos als Open-Source unter der GPL-Lizenz verfügbar³⁸. Da in WEKA viele Klassifikationsalgorithmen und Filter zur Vorverarbeitung der Daten implementiert sind und die Bedienung sehr einfach und intuitiv ist, wurde WEKA vor allem dazu verwendet, explorativ einen ersten Blick auf die Klassifikationsleistung der verschiedenen Verfahren für die hier vorliegende Aufgabenstellung zu werfen. In kurzer Zeit ließen sich so vorab tausende Experimente mit vielen verschiedenen Algorithmen und Parametrisierungen durchführen. Dies führte dann zu dem Entschluss, den CART-Algorithmus zum Herzstück des hier beschriebenen Gegnermodells zu machen. Für die spätere Implementierung des Gegnermodells wurde die WEKA-API verwendet.

In [WFH11] wird sowohl eine Einführung in WEKA gegeben, als auch ein sehr guter Überblick über Techniken und Algorithmen des maschinellen Lernens und Data-Minings.

3.4.2 RapidMiner

Genau wie WEKA ist *RapidMiner* eine Sammlung von Algorithmen und Verfahren des maschinellen Lernens und Data-Minings. Es ist als Open-Source unter der AGPL-Lizenz verfügbar³⁹. Ursprünglich vom *Lehrstuhl für künstliche Intelligenz an der Technischen Universität Dortmund* entwickelt, wird RapidMiner nun von der Firma *Rapid-I* vertrieben und betreut. Der Unterschied zu WEKA liegt hauptsächlich in der grafischen Benutzeroberfläche und in einer größeren Anzahl der implementierten Algorithmen. Beispielsweise sind die meisten in WEKA enthaltenen Verfahren in RapidMiner in der WEKA-Implementierung enthalten. In RapidMiner lassen sich selbst komplexeste Prozesse bzw. Experimente per

³⁸ Für diese Arbeit wurde WEKA in der Version 3.6 verwendet und ist unter <http://www.cs.waikato.ac.nz/ml/weka/> [Stand: Juli 2012] erhältlich.

³⁹ RapidMiner ist unter <http://rapid-i.com/content/view/181/196/> [Stand: Juli 2012] erhältlich. Zur Durchführung der Experimente wurde Version 5.2 verwendet.

Drag&Drop nach einer gewissen Einarbeitungszeit relativ einfach „zusammenziehen“. Die Möglichkeit, Zwischenergebnisse und Modelle zur Laufzeit der Prozesse zu speichern, machen RapidMiner zu einer idealen Umgebung, um Experimente im Bereich des maschinellen Lernens durchzuführen und auszuwerten. Aus diesem Grunde wurde RapidMiner zur Durchführung der hier vorgestellten Experimente verwendet.

3.4.3 Poker Academy Pro 2

Poker Academy Pro 2 ist eine Software, die es erlaubt, gegen verschiedene Pokersysteme der CPRG, die vor allem in [Bil06] beschrieben sind, anzutreten. Es existiert darüber hinaus eine API, die es ermöglicht, eigene Pokersysteme zu implementieren und diese gegen die in *Poker Academy* enthaltenen Pokersysteme spielen zu lassen. Da diese Systeme und die verwendeten Ansätze sehr gut dokumentiert sind, ist *Poker Academy* die ideale Testumgebung für das Heads-Up Limit Spiel⁴⁰ um eigene Systeme zu testen. Die Spiele lassen sich beobachten und es besteht die Möglichkeit, dass mit offenen Karten gespielt wird⁴¹. Dies hat den Vorteil, dass sich bestimmte negative Eigenschaften eines selbst erstellten Pokersystems viel leichter identifizieren lassen als etwa nur durch Analyse der Gewinne und Verluste. *Poker Academy Pro* wurde von der Firma Biotoools Inc. entwickelt.

3.4.4 Java/Eclipse

Die Softwarekomponenten dieser Arbeit wurden mit *Java*⁴² 6 unter der Entwicklungsumgebung *Eclipse*⁴³ in der Helios-Distribution entwickelt.

⁴⁰ Es existieren zwar auch Computer-Gegner für das No Limit Spiel und für die Mehrspieler-Variante, doch sind diese relativ schwach im Vergleich zu den Heads-Up Limit Gegnern.

⁴¹ „Offen“ bedeutet dabei, nur für den Zuschauer sichtbar. Die Computer-Spieler können nicht auf diese Information zugreifen.

⁴² Java steht unter <http://www.java.com/de/download/> [Stand: Juli 2012] zum Download bereit.

⁴³ Eine aktuelle Version von Eclipse findet sich unter <http://www.eclipse.org/downloads/> [Stand: Juli 2012].

3.5 FAZIT

Die Modellierung der Gegenspieler im Texas Hold'em stellt für menschliche Spieler eine große Herausforderung dar. Mindestens genauso groß ist aber auch die Herausforderung, ein gutes Gegnermodell zu entwickeln, das in einer Spielbaumsuche zum Einsatz kommt. Ein Gegnermodell hat in der Spielbaumsuche zwei Aufgaben: Zum Einen die Bereitstellung von Wahrscheinlichkeiten für die Aktionen an den gegnerischen Entscheidungsknoten, und zum Anderen eine Approximation der Erwartungswerte an den Spielendknoten.

Das Teilmodell zur Modellierung der gegnerischen Aktionen, das im folgenden Kapitel analysiert wird, gliedert sich dabei in drei separate, von CART aufgebaute, Klassifikationsbäume. Die hier verwendeten Merkmalsvektoren umfassen 22 Merkmale, sowie das Label der Klasse. Die Datenbasis dieser Arbeit umfasst 66.886 Aktionen zehn verschiedener Spieler, die aus 400.000 Spielen verschiedener Spieler gewonnen wurden.

Ein kurzer Vergleich der Klassifikationsleistung des CART-Algorithmus zu einem KNN zeigt, dass beide Verfahren hinsichtlich der KKR und des RMSE vergleichbar sind. Dies ist besonders interessant, da Arbeiten existieren, die zeigen, dass sich KNNs zur Modellierung gegnerischer Aktionen hinsichtlich der KKR gut eignen. In Bezug auf die Geschwindigkeit und die Stabilität zeigt CART sich dem hier verwendeten relativ simplen KNN überlegen.

4 ANALYSE UND BEWERTUNG DES GEGNERMODELLS

„Ein Abend, an dem sich alle Anwesenden völlig einig sind, ist ein verlorener Abend.“

Albert Einstein

In diesem Kapitel wird die Performanz des CART-Algorithmus für die vorliegende Aufgabenstellung analysiert. Insbesondere wird untersucht, welche Vorhersagegenauigkeit sich für die gegnerischen Aktionen maximal erzielen lässt und wie schnell diese erreicht werden kann. Es wird geprüft, ob sich allgemeine Aussagen darüber treffen lassen, wie viele Trainingsbeispiele benötigt werden, sodass der CART-Algorithmus eine gute Klassifikationsleistung erbringt. Des Weiteren werden bestimmte Eigenschaften und Einschränkungen von CART im Kontext der hier bearbeiteten Aufgabe herausgearbeitet. Die hier teilweise anhand einzelner Spieler und Setzrunden vorgestellten Ergebnisse sind exemplarisch zu verstehen und werden durch weitere Daten, die sich auf der beiliegenden CD-ROM befinden, ergänzt.

4.1 MAXIMAL ERREICHBARE VORHERSAGEGENAUIGKEIT

Um abzuschätzen, welche Vorhersagegenauigkeit sich bezüglich der KKR und des RMSE durch den CART-Algorithmus mit der hier vorgeschlagenen Modellierung der Eingabevektoren für die einzelnen Spielrunden erzielen lässt, wurde für jeden Spieler für die drei Setzrunden jeweils eine 10-fache stratifizierte Kreuzvalidierung durchgeführt. Dies beinhaltet zwar eine gewisse Abstraktion von der tatsächlichen Problemstellung, da die Reihenfolge der gegnerischen Aktionen nicht beachtet wird, dennoch liefert dieser pragmatische Ansatz eine Übereinstimmung mit den in Abschnitt 4.2 vorgestellten Zahlen, die unter realistischeren Bedingungen gewonnen wurden.

Die KKR der Spieler für eine gegebene Setzrunde ist in den Zellen von Tabelle 4 über dem RMSE angeordnet. Die letzte Spalte der Tabelle enthält die mittlere erzielte Vorhersagegenauigkeit über die drei Setzrunden für einen einzelnen Spieler. In der letzten Zeile finden sich die Mittelwerte für die einzelnen Setzrunden über alle Spieler berechnet.

Tabelle 4: KKR und RMSE für alle Spieler und Setzrunden als Ergebnis einer stratifizierten 10-fachen Kreuzvalidierung.

<div> <div>Runde</div> <div>Name</div> </div>	Flop	Turn	River	Ø Spieler
	KKR	KKR	KKR	KKR
	RMSE	RMSE	RMSE	RMSE
Spieler1	0,638	0,705	0,692	0,678
	0,518	0,491	0,51	0,506
Spieler2	0,678	0,646	0,685	0,67
	0,501	0,508	0,48	0,496
Spieler3	0,717	0,687	0,673	0,692
	0,484	0,492	0,48	0,485
Spieler4	0,68	0,655	0,61	0,648
	0,519	0,516	0,527	0,521
Spieler5	0,737	0,682	0,699	0,706
	0,471	0,501	0,513	0,495
Spieler6	0,682	0,705	0,673	0,687
	0,508	0,519	0,487	0,505
Spieler7	0,688	0,704	0,731	0,708
	0,501	0,485	0,462	0,483
Spieler8	0,594	0,65	0,614	0,619
	0,559	0,513	0,513	0,528
Spieler9	0,67	0,725	0,729	0,708
	0,509	0,468	0,463	0,48
Spieler10	0,711	0,653	0,735	0,7
	0,488	0,5	0,459	0,483
Ø Runde	0,679	0,681	0,684	0,682
	0,506	0,499	0,489	0,498

4.1.1 KKR und RMSE nach Setzrunden

Die unter der bestehenden Modellierung erzielte Vorhersagegenauigkeit für die einzelnen Setzrunden über alle Spieler berechnet, liegt bei einer mittleren KKR von 0,682. Für den Flop wurde das schlechteste Ergebnis mit einer KKR von 0,679 erzielt, während die Aktionen für den River mit 0,684 am besten vorhergesagt wurden. Für den Turn wurde eine KKR von 0,681 ermittelt, und somit lagen die mittleren KKR für alle drei Setzrunden in einem Intervall der Breite 0,005. Der Mittelwert des RMSE lag über alle Spieler und Setzrunden berechnet bei 0,498. Wieder lieferte der River mit einem RMSE von 0,489 das beste und der Flop mit 0,506 das schlechteste Ergebnis. Der RMSE für den Turn lag bei 0,499. Das Intervall, in dem die erreichten mittleren RMSE liegen, besitzt die Breite 0,017. Die in Tabelle 4 dargestellten Daten deuten darauf hin, dass sich die Güte der Vorhersagegenauigkeit für die einzelnen Setzrunden nur minimal unterscheidet.

4.1.2 KKR und RMSE nach Spielern

Bei der Betrachtung der Vorhersagegenauigkeit für die einzelnen Spieler zeichnet sich ein anderes Bild als für die Setzrunden. Für vier Spieler wird eine KKR von mindestens 0,7 erreicht und für vier weitere Spieler wird der Wert 0,67 erreicht bzw. überschritten. Für Spieler4 und Spieler8 werden deutlich schlechtere Werte erzielt. Spieler4 wird mit einer KKR von 0,648 über alle Setzrunden modelliert, bei einem RMSE von 0,521. Während der Flop für Spieler4 noch relativ gut mit einer KKR von 0,68 und einem RMSE von 0,519 vorausgesagt wird, beträgt die KKR für den Turn 0,65 und für den River lediglich 0,61.

Für Spieler8 wird insgesamt das schlechteste Ergebnis erzielt. Über alle Runden betrachtet werden eine KKR von nur 0,619 und ein RMSE von 0,528 erreicht. Der Flop liefert dabei die schlechteste KKR und den schlechtesten RMSE aller Spieler bei Betrachtung aller Setzrunden mit Werten von 0,594 für die KKR bzw. 0,559 für den RMSE.

Der Grund für dieses schlechte Ergebnis scheint auf die Tatsache hinzudeuten, dass die Mechanismen, die den Entscheidungen von Spieler4 und Spieler8 zugrunde liegen, nicht ausreichend von den in dieser Arbeit gewählten Eingabeparametern abgebildet werden. Eine zu geringe Anzahl an Trainingsdaten kommt als Erklärungsversuch mit hoher Wahrscheinlichkeit nicht in Frage, da beispielsweise für den Flop für vier Spieler zum Teil

deutlich weniger Trainingsdaten zur Verfügung standen als für Spieler8, diese aber wesentlich bessere Ergebnisse lieferten.

Über alle Setzrunden wird das beste Klassifikationsergebnis für Spieler9 mit einer KKR von 0,708 und einem RMSE von 0,48 erzielt. Die beste KKR und der niedrigste RMSE für eine einzelne Runde wurden für Spieler10 auf dem River mit 0,735 bzw. 0,459 ermittelt.

4.1.3 Fazit

Für die einzelnen Setzrunden wurden hinsichtlich der KKR und des RMSE fast identische Ergebnisse erzielt. Dies lässt den Schluss zu, dass die hier vorgeschlagene Modellierung sich für alle drei Setzrunden in gleicher Weise eignet.

Auf der anderen Seite wurden für die einzelnen Spieler größere Schwankungen in der Vorhersagegenauigkeit beobachtet. Die Werte der KKR lagen zwischen 0,594 und 0,735, wohingegen für den RMSE Werte zwischen 0,459 und 0,559 erzielt wurden. Dies legt die Vermutung nahe, dass weitere Eingabevariablen notwendig sind, die den Kontext der jeweiligen Entscheidungssituationen noch besser erfassen, da jeder Spieler über seine individuellen Präferenzen und Mechanismen verfügt, um seine nächste Entscheidung zu treffen. Zum Teil spielt hier sicherlich auch ein möglicherweise irrationales oder auch täuschendes Verhalten eine Rolle, das eine exakte Modellierung erschwert und bei den verschiedenen Spielern mehr oder weniger ausgeprägt ist.

4.2 ANPASSUNGSGESCHWINDIGKEIT

Ein zentraler Punkt dieser Arbeit ist die Frage, wie viele Datensätze der CART-Algorithmus benötigt, um eine verlässliche Modellierung beliebiger Gegner zu gewährleisten. Dabei geht es vor allem um eine erste grobe Approximation der Größenordnung der benötigten Beobachtungen. Um diese Abschätzung vorzunehmen, wurden verschiedene Experimente durchgeführt, die sich in zwei Gruppen einteilen lassen. Zum Einen wurde die Klassifikationsleistung hinsichtlich der KKR und des RMSE für eine wachsende Trainingsmenge betrachtet. Die zweite Gruppe von Experimenten sollte die

Klassifikationsleistung von Bäumen zueinander in Beziehung setzen, die aufgrund von Trainingsmengen konstanter aber verschiedener Größen erstellt wurden.

4.2.1 Wachsende Trainingsmenge

Dieser Abschnitt soll eine erste grobe Antwort auf die Frage liefern, wie viele Trainingsdaten CART ungefähr benötigt, um eine verlässliche Gegnermodellierung zu erbringen. Aus diesem Grunde wurde ein Versuchsaufbau gewählt, der einem Einsatz von CART in der Praxis entspricht. Hierzu wurden die Trainingsdaten für die betrachteten Spieler in der chronologischen Reihenfolge betrachtet, in der diese beobachtet wurden. Betrachtet wurden jeweils abgeschlossene Datensätze, d.h. die Daten eines Spielers einer einzelnen Setzrunde. Es geht hier vielmehr darum, bestimmte Eigenschaften zu beobachten und Größenordnungen zu identifizieren als um eine genaue Quantifizierung, die für die Fokussierung der Zielstellung dieser Arbeit sicher zu umfangreich wäre.

Um das n -te Element eines Datensatzes zu klassifizieren, bzw. Wahrscheinlichkeiten für die einzelnen Aktionen bereitzustellen, wurde jeweils ein Baum anhand der $n-1$ vorausgehenden Beispiele des Datensatzes aufgebaut und die vorausgesagte Klasse mit dem tatsächlichen Label des n -ten Elementes verglichen. Aus diesen Daten ließen sich dann die KKR und der RMSE an der n -ten Position des Datensatzes errechnen. Zudem sind in den Schaubildern auch die KKR bzw. der RMSE für verschiedene Fenstergrößen zu sehen.

Problematisch ist hierbei, dass sich die Trainingsdaten für die betrachteten Spieler aus Spielen gegen viele verschiedene Gegner zusammensetzen, bei denen möglicherweise komplett unterschiedliche Strategien zum Einsatz kamen. Dennoch soll hier dieser pragmatische Ansatz gewählt werden.

Abbildung 4.1 zeigt das Ergebnis des oben beschriebenen Versuchsaufbaus für Spieler8 auf dem Flop. Dies ist der Datensatz, für den bei der 10-fachen stratifizierten Kreuzvalidierung (siehe Tabelle 4) das schlechteste Klassifikationsergebnis aller hier betrachteten Datensätze erzielt wurde. Die orangefarbene Linie repräsentiert die KKR für alle bis zum jeweiligen Punkt auf der x-Achse klassifizierten Beispiele. Die anderen Linien stellen jeweils die KKR für die letzten 20, 50 und 100 klassifizierten Elemente dar.

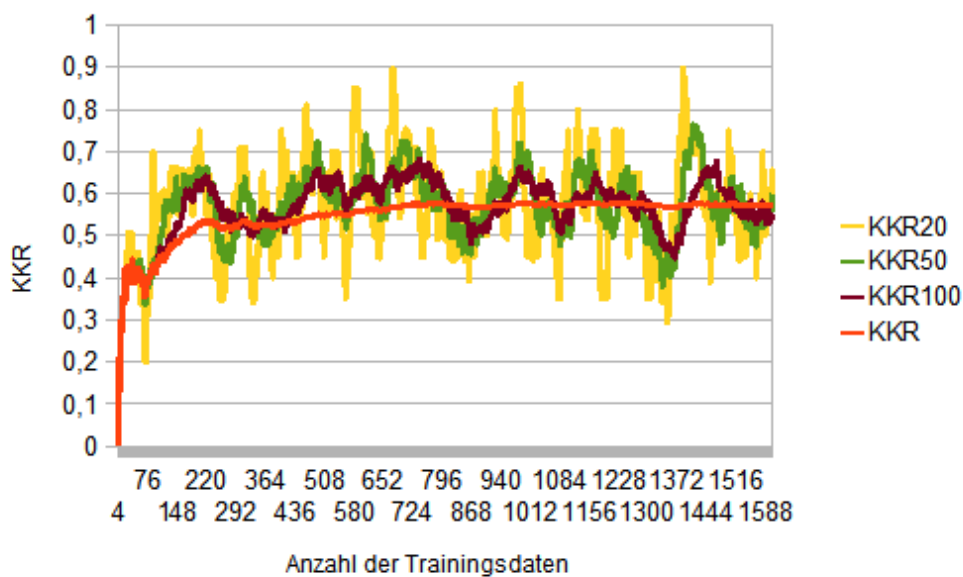


Abbildung 4.1: Lernkurve der KKR bei wachsender Größe der Trainingsmenge für Spieler8 auf dem Flop.

Nach Klassifikation aller Beispiele liegt die mittlere KKR bei 0,574 und somit 0,02 unter der KKR, die bei der 10-fachen Kreuzvalidierung erzielt wurde.

Abbildung 4.1 zeigt, dass CART bereits für ungefähr 100 Trainingsbeispiele relativ gute Vorhersagen bezüglich der KKR macht. Die lilafarbene Linie stellt die KKR für die Fenstergröße 100 dar und erreicht bereits bei einer Größe der Trainingsmenge von 200 den Wert 0,63. Dies bedeutet veranschaulicht, dass beginnend bei einer Trainingsmenge der Größe 100 die 100 folgenden Beispiele bei wachsender Trainingsmenge mit einer KKR von 0,63 klassifiziert werden.

Die orangefarbene Linie erreicht an Position 701 auf der x-Achse das erste Mal eine KKR von 0,57 und scheint sich danach nicht mehr nennenswert zu verändern.

Abbildung 4.2 zeigt das oben beschriebene Experiment unter Betrachtung des RMSE.

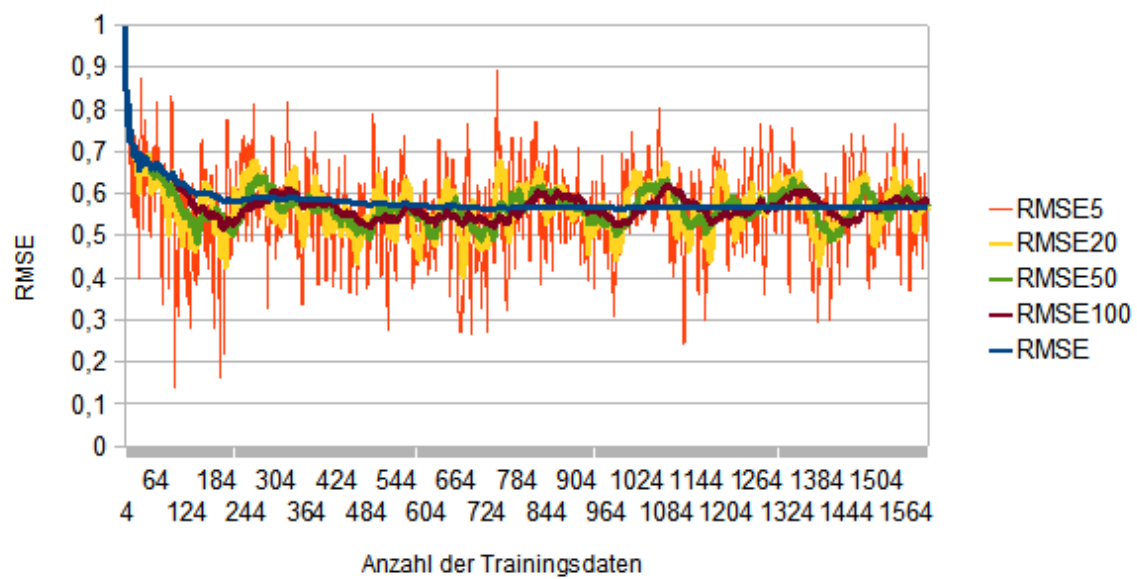


Abbildung 4.2: Lernkurve des RMSE bei wachsender Größe der Trainingsmenge für Spieler8 auf dem Flop.

Nach Klassifikation aller Daten beträgt der RMSE 0,569 und liegt damit 0,01 über dem Wert aus Tabelle 4. Für Position 201 auf der x-Achse erreicht der RMSE der Fenstergröße 100 seinen niedrigsten Wert mit 0,516.

In Abbildung 4.3 wird dasselbe Experiment unter Betrachtung der KKR für Spieler7 auf dem River dargestellt. Für die Kreuzvalidierung wurde das sehr gute Ergebnis einer KKR von 0,731 und einem RMSE von 0,462 erzielt. Analog für den Datensatz von Spieler8 für den Flop liefert CART bereits ab ungefähr 100 Beispielen gute Ergebnisse. Dies verdeutlicht die lilafarbene Linie, die die KKR der Fenstergröße 100 (KKR100) repräsentiert. An Position 200 auf der x-Achse liegt sie bei 0,74. An Position 220 erreicht sie 0,78. Das globale Maximum von KKR100 liegt bei 0,82. Erwähnenswert ist die Tatsache, dass die KKR, die in diesem Experiment nach Klassifikation aller Beispiele erreicht wurde, bei 0,723 liegt und der erzielte RMSE bei 0,474.

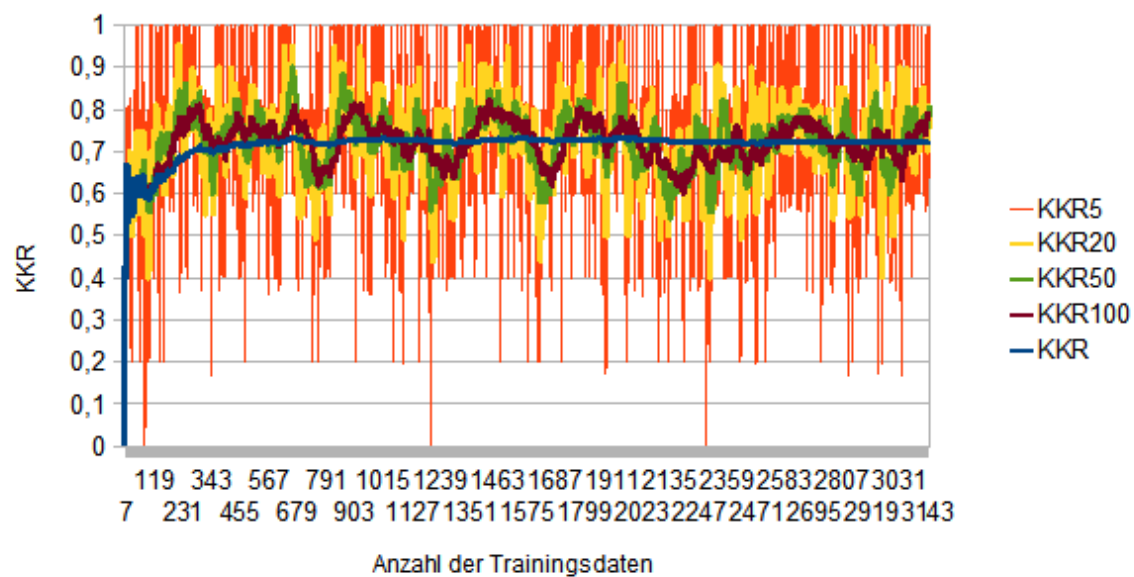


Abbildung 4.3: Lernkurve der KKR bei wachsender Größe der Trainingsmenge für Spieler7 auf dem River.

Unter realistischen Bedingungen sind die erzielte KKR und der RMSE mit den Ergebnissen der 10-fachen stratifizierten Kreuzvalidierung vergleichbar. Akzeptable Werte scheinen sich bereits ab Trainingsmengen von ca. 100 Beispielen einzustellen und sind möglicherweise dadurch erklärbar, dass viele Situationen existieren (d.h. Kombinationen von Setz-Sequenzen und/oder Gemeinschaftskarten), die in gewisser Weise von den Spielern als ähnlich eingestuft werden.

4.2.2 Konstante Größe der Trainingsmenge

In diesem Abschnitt soll untersucht werden, inwiefern sich die Größe der Trainingsmenge, anhand derer CART den Klassifikationsbaum aufbaut, auf die Klassifikationsleistung auswirkt. Hierzu wird eine ähnliche Versuchsanordnung wie im vorangehenden Abschnitt gewählt, allerdings mit konstanten Größen der Trainingsmengen. Wieder sind die Daten chronologisch geordnet. Um die Klasse eines gegebenen Beispiels vorauszusagen, werden aber nicht mehr alle Daten verwendet, die chronologisch vor dem aktuell zu klassifizierenden Beispiel liegen, sondern lediglich die letzten k Daten, wobei k der Größe der jeweils verwendeten Trainingsmenge entspricht.

Abbildung 4.4 zeigt das Ergebnis des Experiments hinsichtlich des RMSE für eine Fenstergröße von 100. Die Größen der Trainingsmengen wurden auf 250, 1000 und 5000 festgesetzt. Betrachtet wurde der Datensatz von Spieler9 auf dem Turn. Für diesen aus 10609 Beobachtungen bestehenden Datensatz wurde in der Kreuzvalidierung eine KKR von 0,725 erreicht, bei einem RMSE von 0,468. Abbildung 4.4 legt nahe, dass Bäume, die anhand von größeren Trainingsmengen aufgebaut werden, generell bessere Ergebnisse erzielen.

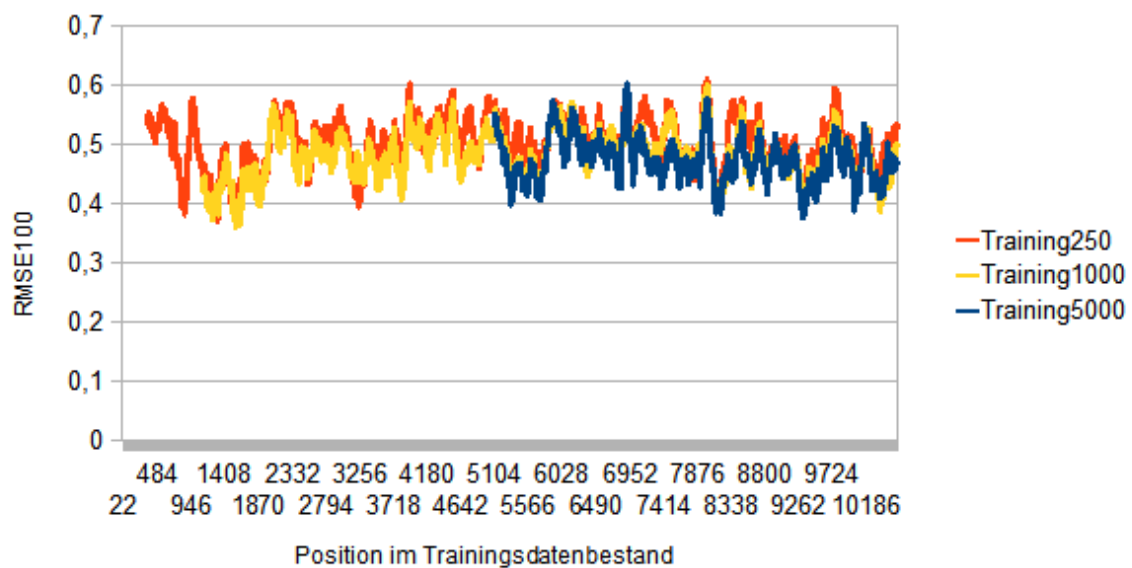


Abbildung 4.4: Darstellung der RMSE100-Linien für verschiedene Größen der Trainingsmengen für Spieler9 auf dem Turn.

Dies lässt sich quantitativ durch einen Vergleich der insgesamt erreichten RMSE und KKR der Bäume mit den Trainingsmengen verschiedener Größen untermauern. Die Berechnung dieser RMSE erfolgte nur an den Beispielen, für die eine Klassifikation aller drei betrachteten Bäume vorlag.

Für eine Größe der Trainingsmenge von 5000 wurde eine KKR von 0,719 erzielt, bei einem RMSE von 0,473. Demgegenüber stehen eine KKR von 0,706 und ein RMSE von 0,485 für eine Trainingsmenge mit 1000 Elementen. Deutlich schlechter präsentiert sich das Ergebnis für 250 Trainingsdaten. Hier wurden eine KKR von 0,685 und ein RMSE von 0,507 erzielt.

Dieses Ergebnis lässt vermuten, dass sich größere Trainingsmengen positiv auf die Klassifikationsleistung auswirken. Dies scheint zunächst im Widerspruch zu den Beobachtungen aus Abschnitt 4.2.1 zu stehen, in dem gezeigt wurde, dass CART bereits mit 100 Trainingsbeispielen gute Ergebnisse liefert. Eine plausible Erklärung hierfür ist die Tatsache, dass durch eine große Menge von Trainingsdaten viele Beobachtungen zum Aufbau des Klassifikationsbaums zur Verfügung stehen, die relativ selten sind, beispielsweise eine ungewöhnliche Setz-Sequenz und/oder bestimmte Kombinationen der Gemeinschaftskarten. Ohne diese seltenen Trainingsdaten wird auch der Klassifikationsbaum nur unzureichend in der Lage sein, ähnliche Instanzen verlässlich zu klassifizieren.

4.2.3 Fazit

Dieser Abschnitt hat gezeigt, dass das Modell unter realistischen Bedingungen eine Klassifikationsleistung erbringt, die der vergleichbar ist, die bei einer 10-fachen stratifizierten Kreuzvalidierung ermittelt wurde. Bereits bei einer Trainingsmenge von ungefähr 100 Beispielen scheinen akzeptable Vorhersageergebnisse erzielbar.

Auf der anderen Seite wurde beobachtet, dass Modelle, die an einer größeren Trainingsmenge erstellt wurden, besser abschneiden, als solche, die anhand einer kleineren Datengrundlage aufgebaut wurden.

4.3 KLASSIFIKATIONSGÜTE DER EINZELNEN AKTIONEN

Ein weiterer wichtiger Aspekt der Klassifikationsleistung von CART vor dem Hintergrund dieser Arbeit ist die Voraussagequalität der einzelnen Klassen. Für die drei Klassen *CHECK/CALL*, *BET/RAISE* und *FOLD* soll hier ein Blick auf die Sensitivität und den positiven Vorhersagewert (PPV) geworfen werden. Die Sensitivität beschreibt den Anteil der korrekt einer Klasse zugeordneten Instanzen an der Gesamtheit der Instanzen dieser Klasse. Demgegenüber drückt der PPV den Anteil der korrekt einer Klasse zugewiesenen Beispiele an der Gesamtheit der dieser Klasse zugewiesenen Beispiele aus (siehe 2.2.2). Tabelle 5 gibt

einen Überblick über die Sensitivität und den PPV für die vorherzusagenden Klassen aller Spieler auf dem River.

Tabelle 5: Übersicht über die Sensitivität und den PPV für die drei Zielklassen aller Spieler auf dem River.

Name \ Klasse	<i>CHECK/CALL</i>	<i>BET/RAISE</i>	<i>FOLD</i>
	Sensitivität	Sensitivität	Sensitivität
	PPV	PPV	PPV
Spieler1	0,8155	0,6626	0,0278
	0,6786	0,72	0,50
Spieler2	0,7549	0,6475	0,4848
	0,718	0,687	0,5333
Spieler3	0,8448	0,4382	0,3253
	0,693	0,6742	0,4909
Spieler4	0,7158	0,6302	0,833
	0,6051	0,6473	0,2917
Spieler5	0,8663	0,6192	0,0388
	0,6874	0,7353	0,5
Spieler6	0,685	0,6395	0,7256
	0,648	0,7106	0,6685
Spieler7	0,8286	0,6728	0,3642
	0,7339	0,7602	0,5584
Spieler8	0,6801	0,6541	0,0891
	0,6419	0,599	0,2727
Spieler9	0,7717	0,7577	0,3632
	0,7214	0,757	0,5995
Spieler10	0,8158	0,6117	0,6387
	0,7639	0,6396	0,7835
Ø Sensitivität ungew.	0,778	0,633	0,428
Ø PPV ungew.	0,684	0,693	0,52
Ø Sensitivität gew.	0,785	0,69	0,451
Ø PPV gew.	0,757	0,729	0,559

In den Zellen der Tabelle ist die Sensitivität über dem PPV angeordnet. Die letzten beiden Zeilen enthalten die Mittelwerte dieser beiden Maßzahlen. In der vorletzten Zeile findet sich der ungewichtete Mittelwert, d.h. jeder Spieler geht mit dem gleichen Gewicht in die Berechnung ein. Die letzte Zeile der Tabelle enthält das gewichtete Mittel, d.h. die Werte jedes Spielers sind in Abhängigkeit der Anzahl der Daten dieses Spielers gewichtet.

Die Tabelle verdeutlicht, dass die Instanzen der Klasse *FOLD* generell am schlechtesten erkannt wurden. Betrachtet man das gewichtete Mittel, so wurden lediglich 45,1% der mit *FOLD* gelabelten Beispiele korrekt vorausgesagt und nur in 55,9% aller Fälle, in denen *FOLD* vorhergesagt wurde, entsprach dies der tatsächlichen Klasse. Für Spieler1 wird die schlechteste Sensitivität bezüglich der Klasse *FOLD* erzielt mit lediglich 2,78%, dies entspricht nur einem Treffer von 36 Instanzen, die das Label *FOLD* tragen.

Im Gegensatz dazu erreicht die mittlere gewichtete Sensitivität der Klasse *CHECK/CALL* einen Wert von 0,785, der gewichtete PPV liegt bei 0,757.

Betrachtet man das gewichtete Mittel, wird für die Klasse *CHECK/CALL* die beste Sensitivität und der beste PPV erreicht. Für *FOLD* liegen beide Werte deutlich unter denen der Klasse *BET/RAISE*, die sich im Mittelfeld befindet.

Aus den beobachteten Werten lassen sich direkt einige Voraussagen für den Einsatz in der Spielbaumsuche machen. Ein gewichteter PPV von 0,559 bedeutet, dass ungefähr 46% der als *FOLD* klassifizierten Beispiele tatsächlich einer der beiden anderen Klassen angehören. Liefert CART häufig zu hohe Wahrscheinlichkeiten für einen Fold des Gegners, so wird die resultierende eigene Spielweise zu aggressiv. Bei schlechten Händen könnte in der Spielbaumsuche an den eigenen Entscheidungsknoten (siehe 2.1.5.3.1) beispielsweise der Erwartungswert für ein Mitgehen unter dem des Aussteigens aus dem Spiel liegen. Liefert das Gegnermodell dann eine hinreichend hohe Wahrscheinlichkeit für ein Aussteigen des Gegners nach einer eigenen Erhöhung, so wird in der Miximax-Baumsuche der höchste Erwartungswert weiter nach oben propagiert. Der extreme Fall, dass dieses Verhalten direkt am Anfrageknoten auftritt, hat eine direkte Erhöhung zur Folge. Zwar ist Bluffen eine notwendige Komponente des Pokerspiels, doch ist eine übermäßig aggressive Spielweise langfristig eine äußerst schwache Strategie und wird von guten Spielern schnell erfasst.

4.3.1 Fazit

In diesem Abschnitt wurde die Klassifikationsleistung von CART hinsichtlich der Sensitivität und des PPV für die Zielklassen untersucht. Es hat sich gezeigt, dass die Klasse *CHECK/CALL* am besten vorausgesagt wird, wohingegen für die Klasse *FOLD* das schlechteste Ergebnis erzielt wird. Beispielhaft wurde gezeigt, dass sich der niedrige PPV für die Klasse *FOLD* als besonders problematisch erweist, da dieser eine übermäßig aggressive Spielweise zu induzieren vermag.

4.4 ÜBERTRAGBARKEIT

Ein gutes Gegnermodell sollte die Eigenschaft besitzen, Aktionen bisher unbekannter Gegner, über die noch keine Informationen zur Verfügung stehen, möglichst sehr genau vorauszusagen, bzw. Wahrscheinlichkeiten für diese Aktionen bereitzustellen. Da CART und alle anderen Klassifikationsalgorithmen eine gewisse Anzahl an Beobachtungen bzw. Trainingsdaten benötigen, bis ein hinreichend gutes Klassifikationsergebnis erzielt werden kann, sind geeignete Mechanismen notwendig, die zum Einsatz kommen, bis diese Trainingsdaten gesammelt wurden.

Eine Möglichkeit dieses Problem zu umgehen, besteht darin, eine Gleichgewichtsstrategie zum Einsatz zu bringen, wie in Abschnitt 2.1.5.2 beschrieben. Darüber hinaus ist es denkbar, dass bestimmte, zuvor erstellte Entscheidungsbäume die Modellierung des neuen, unbekannten Gegners übernehmen. Dies wirft ein neues Problem auf, nämlich das der Zuordnung eines Gegners in eine bestimmte Kategorie von Spielern. Auf dieses Problem soll hier nicht weiter eingegangen werden.

4.4.1 Experiment Ü1: Spieler→Spieler

An dieser Stelle soll geprüft werden, ob sich Beobachtungen, die für einen Spieler gemacht wurden, als Trainingsgrundlage für einen anderen Spieler eignen. Hierzu wurden alle Datensätze der zehn Spieler für den River verwendet. Das Verfahren, das hierzu angewendet wurde, funktioniert analog zu einer 10-fachen stratifizierten Kreuzvalidierung: Um zu testen, ob sich die Daten von SpielerA eignen, die Aktionen von SpielerB zu modellieren, wurden die

Daten von SpielerA in zehn stratifizierte, disjunkte Untermengen aufgeteilt (siehe 2.2.2.1). Wie bei der herkömmlichen Kreuzvalidierung wurden 10 Entscheidungsbäume an jeweils neun dieser Untermengen aufgebaut. Anstatt diese Bäume an der verbleibenden Untermenge für SpielerA zu testen, wurden diese Bäume am gesamten Datensatz für SpielerB getestet. Anschließend wurde der Mittelwert der resultierenden KKR und RMSE der einzelnen Läufe gebildet. Eine Übersicht über das Ergebnis des Experiments gibt Tabelle 6. Jede Zelle enthält in der oberen Hälfte die KKR und in der unteren den RMSE. Zeile A enthält alle Versuche, bei denen die Daten von SpielerA als Trainingsgrundlage verwendet wurden. Spalte B beinhaltet diejenigen Versuche, für die SpielerB die Testdaten zur Verfügung stellte. In der Diagonalen sind zum Vergleich die Werte aufgeführt, die sich auch in Tabelle 4 für den River finden. Um das Ergebnis des Versuchs ansatzweise zu visualisieren, wurden diejenigen Werte in grün dargestellt, die im Vergleich zu den Ergebnissen der Kreuzvalidierung (siehe Tabelle 4) mindestens genauso gut oder besser sind, während rote Werte schlechtere Ergebnisse repräsentieren. Die Diagonalwerte sind in blau dargestellt.

Tabelle 6: KKR und RMSE als Ergebnis einer abgewandelten 10-fachen Kreuzvalidierung. Training und Test wurden an unterschiedlichen Spielern durchgeführt.

Spieler	1	2	3	4	5	6	7	8	9	10
1	0,692 0,51	0,6628 0,534	0,6299 0,546	0,6057 0,559	0,6949 0,514	0,5601 0,592	0,7128 0,503	0,6088 0,552	0,7047 0,506	0,6577 0,537
2	0,6616 0,51	0,685 0,48	0,6388 0,508	0,5927 0,543	0,6796 0,496	0,6015 0,543	0,707 0,484	0,5876 0,543	0,6953 0,496	0,7202 0,475
3	0,6308 0,505	0,6066 0,513	0,673 0,48	0,608 0,521	0,6142 0,514	0,5932 0,53	0,6657 0,486	0,6232 0,513	0,6261 0,504	0,658 0,489
4	0,6928 0,494	0,6404 0,519	0,6464 0,507	0,61 0,527	0,6635 0,508	0,607 0,522	0,6913 0,499	0,6232 0,511	0,692 0,499	0,6588 0,506
5	0,6977 0,509	0,6698 0,53	0,635 0,542	0,609 0,558	0,699 0,513	0,5666 0,592	0,7196 0,499	0,614 0,551	0,7093 0,504	0,6668 0,533
6	0,5919 0,55	0,585 0,538	0,566 0,563	0,5836 0,545	0,5627 0,554	0,673 0,487	0,6129 0,535	0,5443 0,571	0,6103 0,534	0,6077 0,535
7	0,6924 0,489	0,7027 0,477	0,6524 0,506	0,6108 0,531	0,6996 0,484	0,5975 0,545	0,731 0,462	0,6145 0,528	0,7207 0,473	0,7198 0,47
8	0,6375 0,512	0,5388 0,542	0,6042 0,511	0,6099 0,521	0,6111 0,517	0,5692 0,545	0,6402 0,506	0,614 0,513	0,6372 0,512	0,5625 0,532
9	0,6894 0,48	0,6977 0,485	0,6572 0,501	0,6217 0,517	0,6837 0,484	0,6232 0,527	0,724 0,464	0,613 0,522	0,729 0,463	0,7302 0,473
10	0,6808 0,506	0,707 0,475	0,6515 0,502	0,6056 0,543	0,6996 0,495	0,6064 0,545	0,7326 0,477	0,6104 0,536	0,7211 0,488	0,735 0,459

Zusätzlich sind diejenigen Werte in Zeile A und Spalte B unterstrichen, für deren zugehörige Gütemaße (KKR bzw. RMSE) im gleichen Versuchsaufbau, jedoch mit 35-facher Kreuzvalidierung, durch einen anschließenden t-Test⁴⁴ eine signifikante Abweichung hinsichtlich der Ergebnisse ermittelt wurde⁴⁵.

Insgesamt existieren für die KKR und den RMSE jeweils 90 Kombinationen unter den Spielern. In 15 Fällen wurde durch einen fremden Trainingsdatensatz eine gleich gute oder bessere KKR erzielt als für den jeweiligen eigenen Datensatz. In keinem Fall jedoch konnte dieses bessere Abschneiden, unter Zuhilfenahme einer 35-fachen Kreuzvalidierung, als signifikant betrachtet werden. Für den RMSE wurden 19 Kombinationen ermittelt, die besser waren, davon 7 signifikant.

Von den 75 Fällen, in denen eine schlechtere KKR erzielt wurde, können 49 als signifikant betrachtet werden, und für den RMSE 63 der 71 beobachteten schlechteren Kombinationen.

Diese Beobachtungen lassen den Schluss zu, dass es in der Praxis kaum möglich ist, Modelle, die anhand von einzelnen Spielern aufgebaut wurden, für Gegner zum Einsatz zu bringen, über die bisher noch wenige Informationen zur Verfügung stehen. Diese Aufgabe scheint fast unlösbar, vor allem da überdies auch keine Symmetrie der Übertragbarkeit besteht. Beispielsweise vermag das Modell, das an Spieler10 aufgebaut wurde Spieler5 relativ gut zu modellieren. Das Gegenteil ist nicht der Fall, hier sind die resultierenden KKR und RMSE signifikant schlechter als für den eigenen Datensatz.

⁴⁴ Dieser Test wurde analog zu dem t-Test in Abschnitt 3.3.2 durchgeführt, d.h. für ein Konfidenzniveau von 0,95 unter der Nullhypothese der Gleichheit der Mittelwerte beider Stichproben. Die Stichproben sind hier die Ergebnisse der einzelnen Durchläufe der 35-fachen Kreuzvalidierung. Der Versuchsaufbau findet sich auf der beiliegenden CD-ROM.

⁴⁵ Dies bedeutet nicht, dass diese Signifikanz auch den Werten zu unterstellen ist, die durch eine 10-fache Kreuzvalidierung ermittelt wurden. Vielmehr soll hier die Signifikanz der Kombination von Trainings- und Testspieler dargestellt werden. Dieser Ansatz stellt sozusagen eine Art Hilfskonstruktion dar, wurde hier aber gewählt um die durchgängige Verwendung der 10-fachen Kreuzvalidierung in dieser Arbeit beizubehalten.

4.4.2 Experiment Ü2: Kollektiv→Spieler

Analog zum vorausgehenden Abschnitt zeigt Tabelle 7 das Ergebnis eines weiteren Experiments. Dieses sollte eine erste Antwort auf die Frage geben, ob sich eventuell allgemeingültige Modelle aus einer Menge verschiedener Spieler bilden lassen, die sozusagen einem rationalen Basismodell entsprechen. Hierzu wurden die Daten von neun Spielern als Trainingsdaten verwendet, und die Daten des verbleibenden Spielers wurden zum Test der Performanz des Klassifikators genutzt. Wiederum kam das abgewandelte Verfahren der 10-fachen stratifizierten Kreuzvalidierung zum Einsatz, und die betrachtete Spielrunde war der River. Die erste Zeile der Tabelle zeigt die Ergebnisse aus Tabelle 4 für den betreffenden Spieler. In der zweiten Zeile sind die KKR und der RMSE für das hier durchgeführte Experiment zu sehen. Die Bedeutung der Farben und Unterstreichungen entspricht Tabelle 6.

Tabelle 7: KKR und RMSE für eine 10-fache Kreuzvalidierung. Die zweite Zeile zeigt das Ergebnis für das Training an 9 Spielern und den Test am verbleibenden Spieler.

	Spieler1	Spieler2	Spieler3	Spieler4	Spieler5	Spieler6	Spieler7	Spieler8	Spieler9	Spieler10
KKR	0,692	0,685	0,673	0,61	0,699	0,673	0,731	0,614	0,729	0,735
RMSE	0,51	0,48	0,48	0,527	0,513	0,487	0,462	0,513	0,463	0,459
KKR	0,6954	0,6968	0,6638	0,6267	0,7017	<u>0,6251</u>	0,7272	0,6202	0,7232	0,7246
RMSE	<u>0,479</u>	0,483	<u>0,495</u>	0,511	<u>0,482</u>	<u>0,523</u>	0,463	0,517	<u>0,475</u>	0,47

Lediglich für Spieler6 wurde durch die Bäume, die anhand der Daten der neun anderen Spieler erzeugt wurden, eine schlechtere KKR erzielt, die als signifikant angenommen werden kann. Die übrigen ermittelten KKR sind mit den Werten aus Tabelle 4 vergleichbar, in fünf Fällen sogar minimal besser. Für den RMSE kann in fünf Fällen eine Signifikanz unterstellt werden, von denen in zwei dieser Fälle bessere und dreimal schlechtere Werte erzielt wurden.

Dieses Ergebnis lässt vermuten, dass es möglich und bei Weitem einfacher ist, vorgenerierte Modelle gegen bisher unbekannte Gegner zum Einsatz zu bringen, die anhand einer Trainingsmenge erzeugt wurden, die sich aus den Daten verschiedener Spieler zusammensetzt.

Gewiss ist es wünschenswert, verschiedene Standardmodelle zu erzeugen, die unterschiedliche Spielertypen abbilden, z.B. aggressive oder passive Spieler. Eine Untersuchung, wie diese Modelle zu erzeugen sind und wie sie dann neuen Spielern zugeordnet werden, wäre sicherlich von großem Interesse für weitere Arbeiten.

4.4.3 Fazit

Während Modelle von einzelnen Spielern sich auf andere Spieler nur sehr schlecht übertragen zu lassen scheinen, wurden für Modelle, die an einem Kollektiv von Spielern erstellt wurden, sehr gute Ergebnisse erzielt und sie liefern einen guten Ansatzpunkt für weitere Untersuchungen.

4.5 ZUSAMMENFASSUNG

In diesem Kapitel wurde die Performanz des CART-Algorithmus untersucht und einige wichtige Eigenschaften im Kontext der gegebenen Aufgabenstellung herausgearbeitet. Es hat sich gezeigt, dass CART mit den hier verwendeten Merkmalsvektoren alle drei Spielrunden ähnlich gut zu modellieren vermag, während für die einzelnen Spieler eine größere Bandbreite der Ergebnisqualität beobachtet wurde. Unter der gegebenen Modellierung liefert CART bereits ab ungefähr 100 Trainingsbeispielen eine akzeptable Performanz, jedoch wirkt sich ein größerer Trainingsdatensatz positiv auf die KKR sowie auf den RMSE aus.

Die Qualität der Vorhersagen für die Klassen *CHECK/CALL* und *BET/RAISE* hat sich als weit besser erwiesen als für die Klasse *FOLD*. Der PPV der Klasse *FOLD* liegt deutlich unter den erzielten Werten für die anderen beiden Klassen und hat vermutlich in bestimmten Situationen eine übermäßig aggressive Spielweise zur Folge.

Die Ergebnisse, die bei der Untersuchung der Übertragbarkeit von Gegnermodellen erzielt wurden, sind vielversprechend. Zwar waren Modelle, die anhand eines Spielers erstellt und an einem zweiten Spieler getestet wurden nicht in der Lage, akzeptable Ergebnisse zu erzielen, jedoch lieferten Modelle, die an den Daten mehrerer Spieler erstellt wurden, gute bis sehr gute Werte bezüglich der KKR und des RMSE.

5 EINSATZ IN DER PRAXIS

„Die Praxis sollte das Ergebnis des Nachdenkens sein, nicht umgekehrt.“

Hermann Hesse

In diesem Kapitel wird untersucht, wie gut sich das hier entwickelte Modell zur Vorhersage der gegnerischen Aktionen im praktischen Einsatz in der Miximax-Baumsuche bewährt. Die resultierende Spielstärke ist in hohem Maße von einigen Faktoren abhängig, die durch den gewählten Aufbau des Gesamtsystems, im Folgenden auch *TestBot* genannt, vorgegeben sind (siehe unten). Die hier gemachten Beobachtungen besitzen daher nur bedingte Aussagekraft, jedoch soll in dieser Arbeit nicht auf diesen komplettierenden Aspekt verzichtet werden.

5.1 AUFBAU DES GESAMTSYSTEMS: TESTBOT

Der hier verwendete Algorithmus für die Spielbaumsuche ist der von der CPRG entwickelte Miximax-Algorithmus (siehe 2.1.5.3), wobei für die Bereitstellung der Wahrscheinlichkeiten an den gegnerischen Entscheidungsknoten das hier vorgestellte Modell zum Einsatz kam.

Zur Abschätzung der Gewinnwahrscheinlichkeiten bzw. der Erwartungswerte an den Spielendknoten wurde der CART-Algorithmus in ähnlicher Weise wie für die Vorhersage der gegnerischen Aktionen verwendet. Dieses Teilmodell arbeitet mit drei Zielklassen, die den möglichen Spielausgängen entsprechen: *WIN*, *LOSS* und *TIE*. Für jeden beobachteten Showdown, d.h. das Vergleichen der Karten auf dem River, wird eine Beobachtung gemacht, die zum Training für zukünftige Spiele verwendet wird. Die Merkmalsvektoren für dieses Teilmodell wurden 11-dimensional gewählt. Neun dieser Merkmale ähneln denen zur Vorhersage der gegnerischen Aktionen. Ein weiteres Merkmal ist die Handstärke, d.h. ein

Wert zwischen 0 und 1, der angibt, wie wahrscheinlich momentan die stärkste Hand im Spiel gehalten wird. Das elfte Merkmal trägt das Label der Klasse.

Zwar ist dies ein sehr einfaches Modell, jedoch wurden in Vorabtests Korrektklassifikationsraten von ungefähr 0,8 bei einem RMSE von ca. 0,3 erreicht. Der starke Zusammenhang zwischen der eigenen Handstärke und der Wahrscheinlichkeit, das Spiel zu gewinnen, scheinen diese - auf den ersten Blick - guten Werte zu erklären. Welche Werte sich mit anderen Ansätzen erzielen lassen, lässt sich ohne weitere Untersuchungen jedoch kaum abschätzen.

Die Größe der Trainingsmengen war nicht beschränkt und nach Beendigung jedes Spiels wurden durch CART neue Bäume für die Teilmodelle aufgebaut. Jede Serie von Spielen wurde mit leeren Trainingsmengen gestartet⁴⁶.

Für das Spielen vor dem Flop wurde die Preflop-Komponente von *FellOmen2*⁴⁷ verwendet. Dieses Programm, das von Ian Fellows entwickelt wurde und unter der GPL in der Version 2 frei erhältlich ist, belegte bei der jährlichen Computer-Poker Meisterschaft der AAAI⁴⁸ 2008 den zweiten Platz für die hier betrachtete Pokervariante.

5.2 ALWAYS CALL UND ALWAYS RAISE

Zwei Benchmark-Gegner, die sehr häufig zu einer ersten Abschätzung der Stärke eines Pokersystems herangezogen werden, sind *AlwaysCall* und *AlwaysRaise*. *AlwaysCall* repräsentiert dabei die sehr einfache Strategie immer zu schieben, falls möglich. Bei einer Erhöhung wird *AlwaysCall* immer mitgehen.

⁴⁶ Da CART unter der gewählten Parametrisierung zum Aufbau eines Klassifikationsbaums mindestens 6 Beispiele benötigt, lieferten die Teilmodelle bis zum Erreichen dieser Beobachtungen Standardwerte für die Wahrscheinlichkeiten der gegnerischen Aktionen ($\frac{1}{2}$ bzw. $\frac{1}{3}$ - je nachdem, ob zwei oder drei Aktionen zur Verfügung standen) und der Gewinnwahrscheinlichkeiten ($\frac{1}{2}$).

⁴⁷ Unter <http://www.deducer.org/pmwiki/index.php?n=Main.DownloadINOTAndFellOmen> [Stand: Juli 2012] steht *FellOmen2*, das eine spieltheoretische Lösung darstellt, zum Herunterladen bereit.

⁴⁸ Die Homepage dieses Wettbewerbs ist unter <http://www.computerpokercompetition.org> [Stand: Juli 2012] erreichbar.

AlwaysRaise auf der anderen Seite modelliert eine einfache aggressive Strategie. Ist in einer beliebigen Setzrunde noch kein Einsatz des Gegenspielers erfolgt, so wird dieser von AlwaysRaise getätigt. Jeden Einsatz bzw. jede Erhöhung beantwortet AlwaysRaise mit einer weiteren Erhöhung, falls das Limit für Erhöhungen von drei pro Setzrunde noch nicht erreicht wurde.

Es ist anzumerken, dass diese beiden Benchmark-Gegner den Idealfall für das hier vorgestellte Teilsystem zur Modellierung der gegnerischen Aktionen darstellen. CART kann nur beobachtete Aktionen vorhersagen und prognostiziert daher immer die korrekte Aktion, die mit einer Wahrscheinlichkeit von 1 belegt wird. Insofern stellen diese beiden trivialen Gegner für TestBot eine Art Validierungsmöglichkeit der Teilkomponente zur Ermittlung der Erwartungswerte an den Spielendknoten dar.

Abbildung 5.1⁴⁹ zeigt auf der y-Achse die Menge der von TestBot gewonnenen Small Bets im Verlauf einer Serie von 8317 Spielen gegen AlwaysCall. Die Anzahl der Spiele ist auf der x-Achse aufgetragen⁵⁰. Im Laufe dieser Serie konnte TestBot 7862,5 Small Bets (SB) gewinnen. Dies entspricht 0,95 SB pro Spiel.

⁴⁹ Die Abbildungen in diesem Kapitel sind durch Screenshots von Poker Academy Pro entstanden und daher englisch beschriftet.

⁵⁰ Die Bedeutung der beiden Achsen wird für den weiteren Verlauf von Kapitel 5 beibehalten.

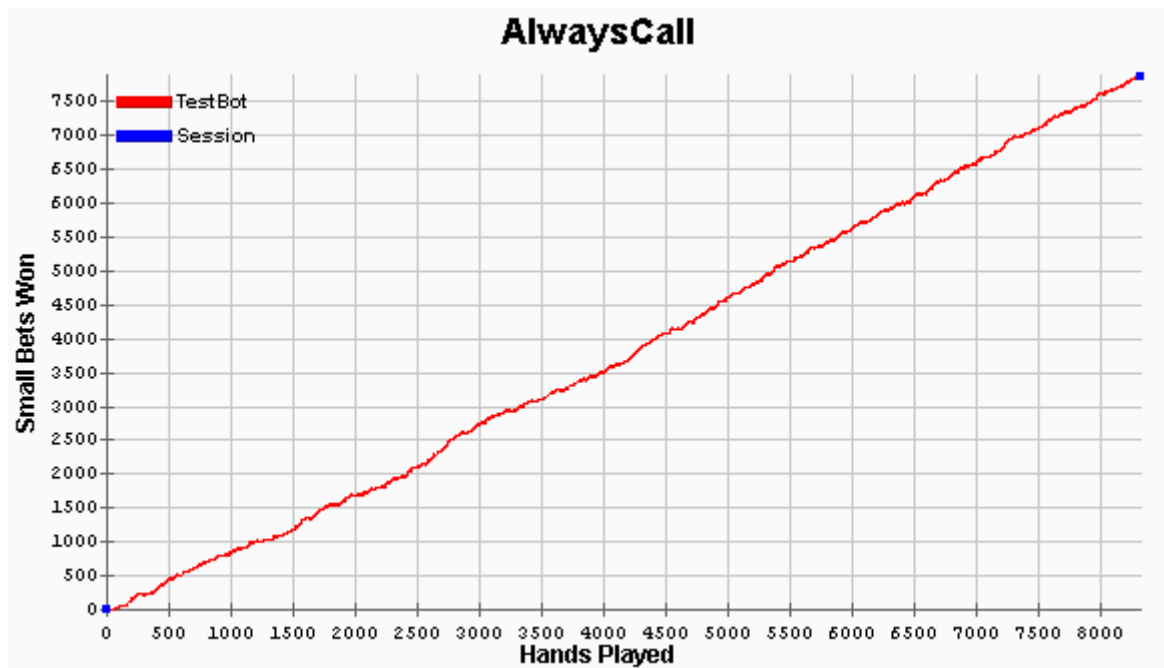


Abbildung 5.1: Eine Serie von 8317 Spielen gegen die Benchmark-Strategie AlwaysCall. TestBot gewinnt insgesamt 7865,5 SB. Das entspricht 0,95 SB pro Spiel.

Zum Vergleich verzeichnete Poki (siehe 2.1.5.1) gegen AlwaysCall 0,51 SB pro Spiel. Verschiedene spieltheoretische Lösungen (siehe 2.1.5.2) konnten zwischen 0,418 und 0,546 SB pro Spiel erzielen. Ein Ansatz unter Verwendung von Miximax erzielte 1,042 SB pro Spiel. Diese Werte beruhen auf Serien von mindestens 40.000 Spielen und finden sich in [Bil06].

In Abbildung 5.2 ist eine Serie von 3524 Spielen gegen AlwaysRaise zu sehen. TestBot konnte im Verlauf dieser Serie 10627 SB gewinnen, d.h. 3,02 SB pro Spiel.

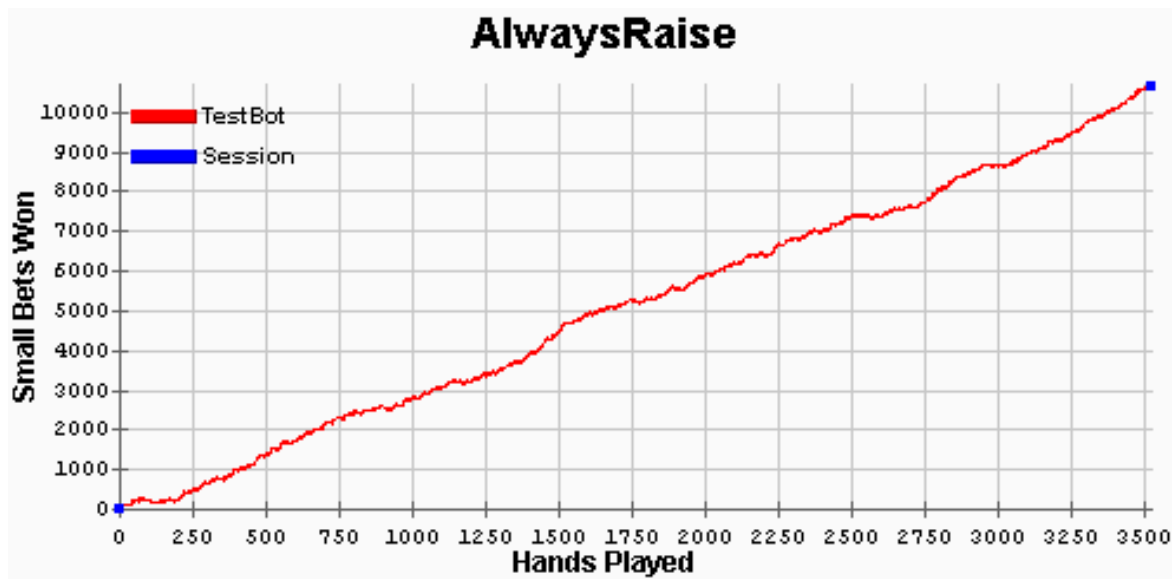


Abbildung 5.2: Eine Serie von 3524 Spielen gegen die Benchmark-Strategie AlwaysRaise. TestBot gewinnt insgesamt 10627 SB, d.h. 3,02 SB pro Spiel.

Poki erzielte gegen diesen Gegner 2,285 SB pro Spiel, wohingegen die spieltheoretischen Lösungen im besten Falle 1,354 SB pro Spiel realisieren konnten. Der Miximax-Ansatz konnte 2,983 SB pro Spiel verbuchen.

5.2.1 Fazit

TestBot vermag die Benchmark-Strategien AlwaysCall und AlwaysRaise überzeugend zu schlagen. Die erzielten Ergebnisse sind mit dem Miximax-Ansatz der CPRG vergleichbar und liegen laut Billings [Bil06] nahe an den theoretisch maximal erzielbaren Werten, die jedoch nicht genannt werden. Die Abweichungen der Ergebnisse von TestBot im Vergleich zu denen des Miximax-Ansatzes der CPRG, scheinen dadurch erklärbar, dass für eine genaue Bewertung weit mehr Spiele erforderlich wären.

Im Verlauf der Spiele gegen die beiden Benchmark-Gegner war die Spielweise von TestBot gut an die beiden Benchmark-Gegner angepasst. Das bedeutet insbesondere, dass keine offensichtlichen Täuschungsversuche wie z.B. Bluffs von TestBot beobachtet werden konnten.

Das relativ schlechte Abschneiden der spieltheoretischen Lösungen gegen die beiden Benchmark-Strategien macht noch einmal deutlich, wie wichtig die Modellierung des Gegners ist, um langfristig einen möglichst maximalen Gewinn zu erzielen.

5.3 POKI

Um einen besseren Blick auf die Spielstärke von TestBot zu werfen, wurde eine Serie von Spielen gegen eine simulationsbasierte Version von Poki durchgeführt. Erwähnenswert ist die Tatsache, dass Poki ebenfalls eine Modellierung des Gegenspielers vornimmt, d.h. seine Strategie in Abhängigkeit der gegnerischen Spielweise ändert. Zwar sind der genaue Aufbau und die verwendeten Parameter in der von Poker Academy Pro bereitgestellten Version nicht bekannt, doch kann Poki als rational spielender Gegenspieler mittlerer Spielstärke verstanden werden und wurde aus diesem Grunde als Gegner gewählt.

Abbildung 5.3 zeigt die Anzahl der von TestBot gegen Poki gewonnenen Small Bets für eine Serie von 18.826 Spielen.

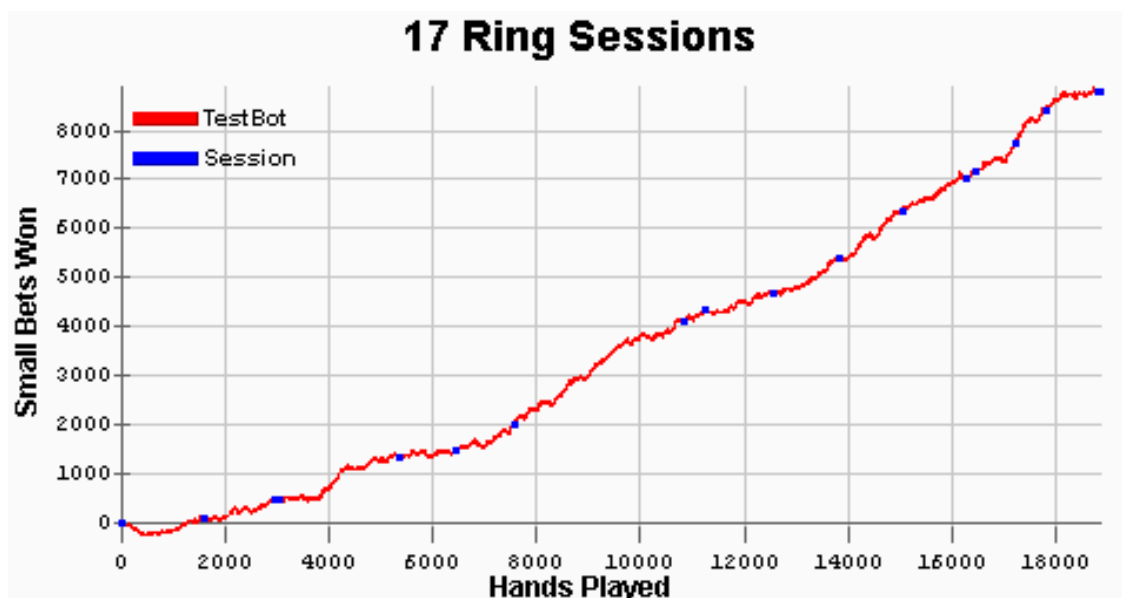


Abbildung 5.3: Im Verlauf einer Serie von 18.826 Spielen vermag TestBot 8811,5 SB zu gewinnen. Dies entspricht 0,47 SB pro Spiel. Beachtenswert ist, dass TestBot im Verlauf der ersten 450 Spiele schlechter zu spielen scheint als Poki und auf ein Minimum von ungefähr -275 SB abfällt.

Insgesamt konnte TestBot 8811,5 SB gewinnen. Dies entspricht 0,47 SB pro Spiel. Die Serie erstreckte sich aus organisatorischen Gründen über 17 Sitzungen bzw. *Sessions*. Das Gegnermodell wurde nach Beendigung einer Session gespeichert und zu Beginn der darauffolgenden Sitzung wieder geladen, sodass es zu nicht zu Informationsverlusten kam.

Auffällig ist, dass TestBot zu Beginn der Serie schlechter zu spielen scheint als Poki und nach ca. 450 absolvierten Spielen bis auf ungefähr -275 SB abrutscht. Erst nach ungefähr 1.300 Händen kann TestBot seine Bilanz wieder auf +/- 0 SB ausgleichen.

Dieses schlechte Abschneiden von TestBot während der Anfangsphase kann durch zwei Faktoren erklärt werden. Zum Einen zeigt eine Analyse der ersten 450 Hände, dass Poki deutlich öfter die beste Hand im Spiel hatte als TestBot. Der Einfluss der erhaltenen Karten kann allerdings nur äußerst schwer abgeschätzt werden. Einen Einblick hierzu liefern Billings und Kan in [BK06]. Der andere erklärende Faktor ist die Tatsache, dass das Gegnermodell ohne Beobachtungen über Poki beginnt und die einzelnen Teilmodelle eine gewisse Anzahl an Beobachtungen benötigen, um eine akzeptable Leistung zu erbringen. Besonders negativ wirkt sich aus, dass nur wenige Spiele mit einem Showdown enden. Dies hat zur Folge, dass die Trainingsmenge für das Teilmodell, das zur Bereitstellung der Gewinnwahrscheinlichkeiten an den Spielendknoten verantwortlich ist, nur sehr langsam wächst. Im Verlauf der 18.826 Spiele wurden 2.886 Showdowns beobachtet, dies entspricht 15,3% aller Spiele. Von den ersten 450 Spielen endeten 99 Spiele (22%) mit einem Showdown.

Diese Beobachtungen machen noch einmal deutlich, wie wichtig Standardmodelle für einen praktischen Einsatz gegen unbekannte Gegner sind. Wie bereits erwähnt, bieten sich hier die generische und die gruppenspezifische Gegnermodellierung an.

Billings berichtet in [BDSB04] von einer Gewinnrate von 0,601 SB des Miximax-Ansatzes gegen Poki. Da aber viele verschiedene Versionen von Poki existieren, ist dieser Wert nicht direkt mit dem hier erzielten Wert vergleichbar. Verschiedene spieltheoretische Lösungen verzeichneten laut [BDSS02] zwischen 0,001 und 0,059 SB pro Spiel.

Im Ganzen betrachtet, scheint die Spielweise von TestBot gut an seinen Gegner angepasst zu sein. Eine offensichtliche Schwäche von TestBot ist allerdings eine vereinzelt auftretende übermäßige Bereitschaft zum Bluffen. Das bedeutet, dass TestBot gelegentlich mit sehr

schwachen Händen unangemessen aggressiv spielt und unnötigerweise viele Small Bets verliert. Dies erklärt sich mit aller Wahrscheinlichkeit dadurch, dass der positive Vorhersagewert für die Klasse *FOLD* relativ schlecht ist und dem Gegner damit eine zu hohe Wahrscheinlichkeit unterstellt wird, aus dem Spiel auszusteigen.

5.3.1 Fazit

TestBot ist in der Lage, Poki deutlich zu schlagen. Zwar ist das Abschneiden von TestBot mit dem Maximax-Ansatz der CPRG aufgrund mangelnder Informationen über die verwendeten Parameter nicht direkt vergleichbar, doch scheint die Größenordnung der gewonnenen SB pro Spiel eine ähnliche zu sein. Als besonders problematisch erwies sich, dass TestBot vereinzelt eine sehr aggressive Spielweise zeigt. Dies wirkt sich negativ auf den langfristigen Erwartungswert aus und ist vermutlich mit dem relativ schlechten PPV der Klasse *FOLD* zu begründen.

5.4 ZUSAMMENFASSUNG UND WEITERE ÜBERLEGUNGEN

Um den CART-Algorithmus für die Modellierung der gegnerischen Aktionen nicht nur aus theoretischer Sicht zu betrachten, wurde ein Pokersystem entwickelt, das den Namen TestBot trägt. Dieses System war in der Lage, die beiden Benchmark-Strategien AlwaysCall und AlwaysRaise überzeugend zu schlagen. Die erzielten Gewinne pro Spiel sind mit denen des Maximax-Ansatzes der CPRG vergleichbar.

Im Einsatz gegen Poki, das als rational spielender Gegner verstanden werden kann, hat sich gezeigt, dass TestBot im Verlauf vieler Spiele deutlich überlegen ist. Problematisch hat sich erwiesen, dass das Gegnermodell von TestBot zu Beginn der Serie über keinerlei Informationen über Poki verfügte und dass insbesondere Beobachtungen über die gegnerischen Karten relativ selten gemacht werden konnten. Für einen praktischen Einsatz, bei dem eine Serie von Spielen möglicherweise nur einige hundert Hände umfasst, scheinen vorgenerierte Standardmodelle unumgänglich zu sein.

Insgesamt hat die Spielweise von TestBot überzeugt. Gegen die beiden Benchmark-Gegner konnten im Wesentlichen keine - ohnehin sinnlosen - Täuschungsversuche seitens TestBot

beobachtet werden. Gegen Poki wurde vor allem die Problematik identifiziert, dass TestBot vereinzelt mit sehr schwachen Händen sehr viele Small Bets verlor, was vermutlich auf den relativ schlechten positiven Vorhersagewert der Klasse *FOLD* zurückzuführen ist.

Abschließend sei bemerkt, dass es sich bei TestBot nicht um ein auf Geschwindigkeit optimiertes System handelt. In der Spielbaumsuche werden im Zusammenspiel mit dem Gegnermodell viele zeitintensive String-Manipulationen für die möglichen Setz-Sequenzen durchgeführt. Dies hat zur Folge, dass TestBot sehr viel Zeit benötigt, um hinreichend viele Spiele zu absolvieren. Gegen Poki dauerte ein Spiel typischerweise zwischen 30 Sekunden und einer Minute. Die Zeit ist abhängig davon, ob TestBot zuerst am Flop zu agieren hatte und ob bestimmte Teilbäume aufgrund einer Wahrscheinlichkeit von 0 nicht durchsucht werden mussten. Gegen AlwaysCall und AlwaysRaise benötigte TestBot deutlich weniger Zeit⁵¹. Dies liegt daran, dass für diese beiden Gegner aufgrund der Wahrscheinlichkeiten von 0 für bestimmte Aktionen, große Teile des Spielbaums nicht bearbeitet werden mussten. Insbesondere die Spiele gegen AlwaysCall konnten vergleichsweise schnell durchgeführt werden, da hier die einzelnen Runden durch ein vorgegebenes Mitgehen von AlwaysCall sehr schnell beendet waren. Für weitere Arbeiten wäre sicherlich eine optimierte Version der Spielbaumsuche wünschenswert. Insbesondere eine Betrachtung der Spielstärke unter Verwendung von Standardmodellen bietet sich als ein interessantes Vertiefungsgebiet an. Hierzu wäre vor allem eine Erweiterung um eine Import- und Filterfunktion wünschenswert, die es gestattet, beispielsweise die hier betrachteten Daten realer Spieler zu nutzen.

⁵¹ Ein Spiel gegen AlwaysCall dauerte typischerweise 5-10 Sekunden, gegen AlwaysRaise ca. 10-25 Sekunden. Für die Spiele gegen die beiden Benchmark-Strategien wurde ungefähr die gleiche Zeit zur Verfügung gestellt. Hieraus erklären sich auch die stark unterschiedlichen Anzahlen der Spiele gegen AlwaysCall und AlwaysRaise.

6 ZUSAMMENFASSUNG UND AUSBLICK

„Die Zukunft soll man nicht voraussehen wollen, sondern möglich machen.“

Antoine de Saint-Exupéry

Die Aufgabenstellung dieser Arbeit war die Analyse und Bewertung des CART-Algorithmus zur Modellierung gegnerischer Aktionen für die 2-Spieler Variante des Texas Hold'em. Hierfür wurden 66.886 Aktionen zehn verschiedener Spieler betrachtet, die auf einer Online-Plattform um Geld gespielt wurden. Im Folgenden werden das Vorgehen und die Ergebnisse dieser Arbeit kurz zusammengefasst. Abschließend erfolgt ein Ausblick auf mögliche zukünftige Entwicklungen und weitergehende Ansätze.

6.1 ZUSAMMENFASSUNG

Aus einem Datenbestand von knapp 400.000 Spielen wurden 10 Spieler aus den 20 Spielern mit den meisten Spielen zufällig ausgewählt. Für die Spiele dieser Spieler wurden unter Beachtung der zeitlichen Reihenfolge, in der die Spiele beobachtet wurden, die Datensätze generiert, die zum Training bzw. zum Testen des CART-Algorithmus verwendet wurden. Aufgrund der verschiedenen a priori-Wahrscheinlichkeiten der Aktionen der einzelnen Spieler für die verschiedenen Setzrunden, wurde die Entscheidung getroffen, für jede Setzrunde ein eigenes Teilmodell zu erzeugen. Die Merkmalsvektoren umfassen das Label und 22 Merkmale, die jede Entscheidungssituation charakterisieren.

In einem explorativen Vergleich zu einem KNN wurde festgestellt, dass CART eine vergleichbare Klassifikationsleistung erbringt, allerdings in wesentlich kürzerer Zeit und ohne die Problematik lokaler Minima oder die Notwendigkeit einer Parametrisierung.

In Kapitel 4 wurde der CART-Algorithmus ausführlich vor dem Hintergrund der vorliegenden Aufgabenstellung getestet. Es wurde gezeigt, dass die verschiedenen Setzrunden ähnlich gut mit einer mittleren KKR von 0,682 modelliert werden, wohingegen sich die Güte der Vorhersagen für die einzelnen Spieler zum Teil deutlich unterscheidet. Dies führte zu der Vermutung, dass weitere Merkmale notwendig sind, die noch mehr Kontext für die Spielsituationen erfassen, um bessere Ergebnisse zu liefern. Die Frage nach einer Mindestgröße für die Trainingsmenge, sodass CART verlässliche Ergebnisse liefert, konnte beantwortet werden. Es hat sich gezeigt, dass CART im Bereich von 100 Trainingsdaten beginnt, verlässliche Vorhersagen zu erzielen. Mehr Trainingsdaten lieferten bessere Ergebnisse, jedoch schien die Verbesserung ab einem gewissen Punkt zu stagnieren.

Die Problematik, dass CART erst ab einer gewissen Anzahl von Trainingsdaten gute Ergebnisse erbringt, führte zu der Frage, ob sich Daten von Spielern als Trainingsdaten für andere Spieler eignen. Für den Fall, dass CART an den Daten eines Spielers trainiert und an einem anderen Spieler getestet wurde, sind die Ergebnisse als unbefriedigend zu bezeichnen. Für den Fall, dass die Daten eines Spielers als Testdaten und die Daten der verbleibenden neun anderen Spieler als Trainingsdaten verwendet wurden, konnten sehr gute Ergebnisse erzielt werden. Vermutlich wurde hier eine Art kollektives, rationales Spielverhalten abgebildet.

Weiterhin wurden die Sensitivität und der positive Vorhersagewert der Zielklassen untersucht. Insbesondere die relativ schlechte Performanz bei der Vorhersage von Folds wurde als mögliche Problematik identifiziert.

Der Einsatz gegen zwei Benchmark-Strategien und das von der CPRG entwickelte Poki sollte den hier vorgestellten Ansatz in der Praxis testen. Dies war insofern problematisch, als dass die gewählten Rahmenbedingungen einen großen Einfluss auf den Ausgang hatten, der nicht zu quantifizieren ist. Zum Einen die verwendete Baumsuch-Variante Miximax, die an den eigenen Spielknoten immer den höchsten Erwartungswert weiter nach oben im Baum propagiert. Zum Anderen wurde zur Abschätzung der Gewinnwahrscheinlichkeiten an den Endknoten des Spielbaumes ein Modell eingesetzt, das dem Modell zur Vorhersage der gegnerischen Aktionen entspricht, allerdings mit leicht abgewandelten Merkmalsvektoren.

Es hat sich gezeigt, dass die Spielweise des resultierenden Gesamtsystems sehr gut an die beiden Benchmark-Gegner angepasst war und die Höhe des Gewinns gegen diese beiden Gegner nahe am theoretisch erzielbaren Maximum lag.

Das Abschneiden gegen Poki kann im Verlauf einer Serie von mehr als 18.000 Spielen mit 0,47 gewonnenen Small Bets pro Spiel als sehr gut bewertet werden, jedoch war TestBot erst nach ungefähr 450 absolvierten Spielen deutlich besser, was durch fehlende Informationen über die Spielweise von Poki zu Beginn der Serie zu erklären ist und die Notwendigkeit von Standardmodellen unterstreicht. Desweiteren scheinen irrationale Entscheidungen, wie z.B. das vereinzelte übermäßig aggressive Bluffen mit sehr schlechten Händen, den langfristigen Erwartungswert stark zu senken.

Nach Abschluss der Arbeit und Betrachtung der gewonnenen Erkenntnisse lässt sich eine Vielzahl an Erweiterungen und Verfeinerungen des hier vorgestellten Ansatzes identifizieren.

6.2 AUSBLICK

Um die Klassifikationsleistung des hier betrachteten Modells zu steigern, ist die Identifikation weiterer Merkmale, die noch mehr Spielkontext bzw. Informationen über den zu modellierenden Spieler abbilden, ein Punkt, dem in Zukunft eine größere Aufmerksamkeit zuteilwerden sollte. Insbesondere Inferenzen über mögliche gegnerische Karten scheinen hierzu besonders geeignet.

Desweiteren sollte die Problematik der Erzeugung von Standardmodellen, um bisher unbekannte Gegner zu modellieren, eingehender beleuchtet werden. Die Ergebnisse von Experiment Ü2 (siehe 4.4.2) scheinen ermutigend. Dies wirft die Frage auf, wie diese Standardmodelle erzeugt werden sollen, bzw. anhand welcher Metriken bestimmte Spielertypen hinreichend schnell identifiziert werden können, sodass das jeweilige Standardmodell korrekt gewählt wird. Van der Kleij wirft in [van10] einen ersten Blick auf diese Problematik für das No Limit Texas Hold'em.

Bei der Verwendung solcher Standardmodelle, stellt sich die Frage, wie die Beobachtungen, die über einen neuen Spieler gemacht werden, in das Modell einfließen sollen. Eine Möglichkeit wäre beispielsweise, ein Standardmodell und ein Modell, das aus den

Beobachtungen über einen relativ unbekannten Spieler erstellt wurde, zu kombinieren. Jedes Modell könnte mit einem gewissen Gewichtungsfaktor in die Abschätzung der gegnerischen Aktionen eingehen, abhängig von der Anzahl der Beobachtungen, die für den zu modellierenden Spieler bereits gemacht wurden. Darüber hinaus ist es denkbar, dass die Trainingsmenge für den zu modellierenden Spieler auf eine gewisse Größe begrenzt wird, sodass nur die n letzten Beobachtungen zum Training des spezifischen Gegnermodells verwendet werden. Das verwendete Standardmodell würde somit eine allgemeine, rationale Spielweise modellieren. Die spezifische Komponente dient dazu, die momentane Strategie des Gegners abzubilden.

Ein perfektes Gegnermodell wird es wohl nie geben, doch aus der Beschäftigung mit dieser Thematik sollten sich durchaus wertvolle Erkenntnisse für die Entwicklung immer stärkerer Pokersysteme gewinnen lassen.

LITERATURVERZEICHNIS

- [Alp08] Ethem Alpaydin. Maschinelles Lernen. Oldenbourg Verlag, München, 2008.
- [BDSB04] D. Billings, A. Davidson, T. Schauenberg, N. Burch, M. Bowling, R. Holte, J. Schaeffer und D. Szafron. Game-tree search with adaption in stochastic imperfect-information games. Hrsg.: H.J. van den Herik, Y. Björnsson und N. Netanyahu. Computers and Games: 4th International Conference, CG'04. Ramat-Gan, Israel. Springer-Verlag GmbH, Bd. 3846 of Lecture Notes in Computer Science, S. 21-34, 2004.
- [BDSS02] D. Billings, A. Davidson, J. Schaeffer und D. Szafron. The challenge of poker. Artificial Intelligence, 134(1-2), S. 201-240, 2002.
- [BFOS84] Leo Breiman, Jerome H. Friedman, Richard A. Olshen und Charles J. Stone. Classification And Regression Trees. Chapman&Hall, New York, 1984.
- [Bil06] Darse Billings. Algorithms and Assessment in Computer Poker. University of Alberta, Edmonton, 2006.
- [BK06] Darse Billings und Morgan Kan. A Tool for the Direct Assessment of Poker Decisions. University of Alberta, Edmonton, 2006.
- [Bur97] M. Buro. The Othello match of the year: Takeshi Murakami vs. Logistello. ICCA Journal, 20(3), S. 189-193, 1997.
- [CBHK02] Nitesh V. Chawla, Kevin W. Bowyer, Lawrence O. Hall und W. Philip Kegelmeyer. SMOTE: Synthetic Minority Over-sampling Technique. Journal of Artificial Intelligence Research, 16, S. 321-357, 2002.
- [CHH02] M. Campbell, A. J. Hoane und F-h Hsu. Deep Blue. Artificial Intelligence, 134(1-2), S. 57-83, 2002.
- [Dav99] Aaron Davidson. Using Artifical Neural Networks to Model Opponents in Texas Hold'em. University of Alberta, Edmonton, 1999.
- [Dav02] Aaron Davidson. Opponent Modeling in Poker: Learning and Acting in a Hostile and Uncertain Environment. University of Alberta, Edmonton, 2002.
- [DBS00] A. Davidson, D. Billings und J. Schaeffer. Improved opponent modeling in poker. International Conference on Artificial Intelligence, ICAI'00, S. 1467-1473, 2000.
- [Fin61] N. V. Findler. Programming Games. Summarized Proceedings of the First Conference on Automatic Computing and Data Processing, Paper B1, 3.3., Australien, 1961.
- [HK06] Jiawei Han und Micheline Kamber. Data Mining. Concepts and Techniques, 2. Auflage, Morgan Kaufmann Publishers, 2006.
- [JM11] Nathalie Japkowicz, und Shah Mohak. Evaluating Learning Algorithms: A Classification Perspective. Cambridge University Press, 2011.
- [Joh07] Michael Patrick Johanson. Robust Strategies and Counter-Strategies: Building a Champion Level Computer Poker Player. University of Alberta, Edmonton, 2007.

- [MMRS06] John McCarthy, Marvin L. Minsky, N. Rochester und C.E. Shannon. A Proposal for the Dartmouth Summer Research Project on Artificial Intelligence. AI Magazine, Bd. 27, 4, 2006.
- [NM44] J. von Neumann und O. Morgenstern. The Theory of Games and Economic Behavior. Princeton University Press, 1944.
- [Pap98] Denis Richard Papp. Dealing with Imperfect Information in Poker. University of Alberta, Edmonton, 1998.
- [Qui86] J. R. Quinlan. Induction of Decision Trees. Machine Learning 1: S. 81-106, 1986.
- [RB92] M. Riedmiller und Heinrich Braun. Rprop - A Fast Adaptive Learning Algorithm. Proceedings of the International Symposium on Computer and Information Science VII, 1992.
- [RN10] Stuart Russell und Peter Norvig. Artificial Intelligence. A Modern Approach, 3. Auflage, Prentice Hall, 2010.
- [RS04] Laura E. Raileanu und Kilian Stoffel. Theoretical Comparison between the Gini Index and Information Gain Criteria. University of Neuchâtel, Computer Science Departement, 2004.
- [Sch97] J. Schaeffer. One Jump Ahead: Challenging Human Supremacy in Checkers. Springer-Verlag, 1997.
- [SM64] J. A. Sonquist und J. N. Morgan. The detection of interaction effects. Institute for Social Research, University of Michigan, Ann Arbor, 1964.
- [Tim04] Roman Timofeev. Classification and Regression Trees (CART) Theory and Applications. Humboldt Universität, Berlin, 2004.
- [TK07] Grigorios Tsoumakas und Ioannis Katakis. Multi-Label Classification: An Overview. International Journal of Data Warehousing & Mining, 2007.
- [van10] A.A.J. van der Kleij. Monte Carlo Tree Search and Opponent Modeling through Player Clustering in no-limit Texas Hold'em Poker. Universität Groningen, Niederlande, Masterarbeit, 2010.
- [Ven06] Joel Veness. Expectimax Enhancements for Stochastic Game Players. The University of New South Wales School of Computer Science and Engineering, Sydney, 2006.
- [WBB09] Kevin Waugh, Nolan Bard und Michael Bowling. Strategy Grafting in Extensive Games. University of Alberta, Edmonton, 2009.
- [WFH11] Ian H. Witten, Eibe Frank und Mark A. Hall. Data Mining. Practical Machine Learning Tools and Techniques. Morgan Kaufmann Publishers, 2011.
- [Wie61] Norbert Wiener. Cybernetics: or Control and Communication in the Animal and the Machine, MIT Press, 1961.
- [Wol95] D. H. Wolpert. The Relationship between PAC, the Statistical Physics Framework, the Bayesian Framework, and the VC Framework. Addison-Wesley, Reading, MA, 1995.

- [WP01] Gary M. Weiss und Foster Provost. The Effect of Class Distribution on Classifier Learning: An Empirical Study. Department of Computer Science, Rutgers University, Technical Report ML-TR-44, 2001